แบบฝึกปฏิบัติ
เริ่มเรียนเขียนโปรแกรม

ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์
จุฬาลงกรณ์มหาวิทยาลัย

v1.0.1

คำนำ

ฉบับ e-book

วิชา ๒๑๐๑๐๑ การเขียนโปรแกรมคอมพิวเตอร์เป็นหนึ่งในวิชาพื้นฐานทางวิศวกรรมศาสตร์ ที่นิสิตชั้นปีที่ ๑ ทุกคนของคณะวิศวกรรมศาสตร์ต้องลงทะเบียนเรียน วัตถุประสงค์หลักของวิชานี้ คือ ให้นิสิตเข้าใจหลักการในการใช้คำสั่งต่าง ๆ ของภาษาโปรแกรม เพื่อเขียนโปรแกรมคอมพิวเตอร์ให้ตรงตามข้อกำหนดที่ได้รับ การเขียนโปรแกรมเป็นความสามารถที่พัฒนาได้มาด้วยการลงมือปฏิบัติด้วยตนเอง เหมือนกับทักษะอื่นทางวิศวกรรมที่จำเป็นต้องฝึก ๆ ๆ ไม่สามารถได้มาด้วยการอ่าน ๆ ๆ

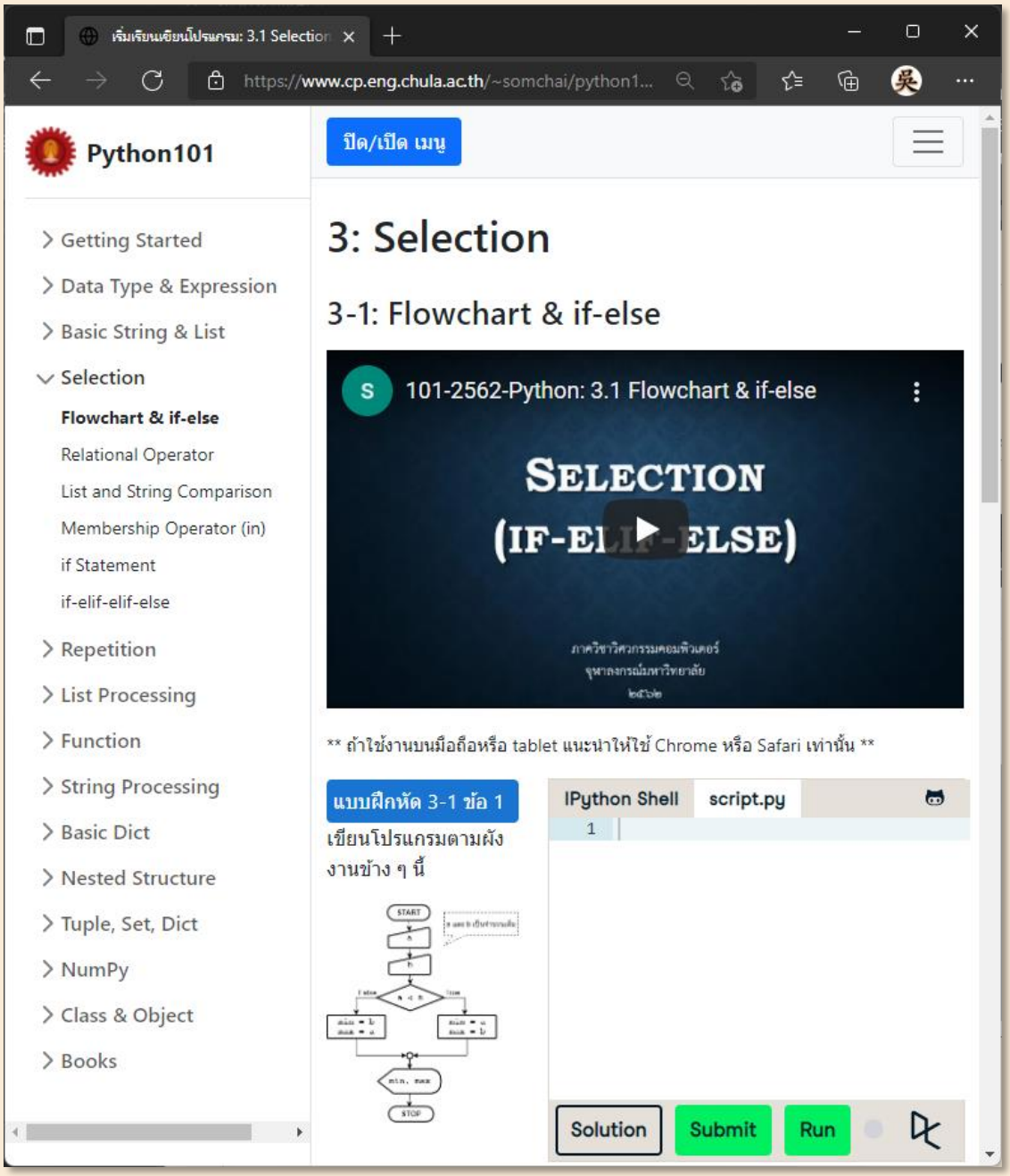
แบบฝึกปฏิบัติเริ่มเรียนเขียนโปรแกรมเล่มนี้รวบรวมโจทย์ปัญหาที่นิสิตต้องฝึกปฏิบัติตลอดภาคการศึกษา (แยกเป็นบท ๆ ตามหัวข้อที่เรียน) และแบบฝึกปฏิบัติเพิ่มเติมอีกจำนวนหนึ่ง (ที่รวบรวมจากโจทย์ทดสอบในอดีต) โดยใช้ระบบตรวจโปรแกรมอัตโนมัติ Grader [1] ในการตรวจสอบความถูกต้องของโปรแกรม หากมีปัญหาใดที่ต้องการคำอธิบายเพิ่มเติม นิสิตสามารถสอบถามปัญหาได้กับอาจารย์และพี่นิสิตช่วยสอนผ่านระบบออนไลน์ที่แจ้งไว้ หากต้องการดูเฉลย ก็สามารถกดที่รูป (ที่มุมขวาบนของโจทย์) เพื่อชมวิดีทัศน์การเขียนโปรแกรมของแต่ละโจทย์ปัญหา[2] ถ้าต้องการทบทวนเนื้อหาของบทใด ก็สามารถกดที่รูป ในหน้าแรกของบทนั้น หรือจะไปที่ https://www.cp.eng.chula.ac.th/~somchai/python101 เพื่อชมวีดีทัศน์บทเรียนทุกบทพร้อมแบบฝึกหัดสั้น ๆ พร้อมตัวตรวจและเฉลย (คำถามจะง่ายกว่าในหนังสือเล่มนี้)

ขอขอบคุณ ผศ. ดร. นัทที นิภานันท์ ผู้ปรับปรุงระบบตรวจโปรแกรม Grader เพื่อใช้ประกอบการเรียนการสอน และการสอบวิชาการเขียนโปรแกรม ขอขอบคุณคณาจารย์และนิสิตช่วยสอนที่ร่วมกันสร้างโจทย์ปัญหา สอน และปรับปรุงวิชาตั้งแต่ปี พ.ศ. ๒๕๕๘ ขอขอบคุณภาควิชาวิศวกรรมคอมพิวเตอร์ที่ให้การสนับสนุนในสารพัดเรื่อง และท้ายสุดที่ต้องขอบคุณ คือ นิสิตคณะวิศวฯ กว่าหลายพันคนที่ขยันหมั่นศึกษาและฝ่าฟันอุปสรรคในการเรียนวิชาพื้นฐานบังคับที่ค่อนข้างไม่คุ้นเคยนี้จนสำเร็จ

รศ. ดร. สมชาย ประสิทธิ์จูตระกูล
(ผู้รวบรวมและเรียบเรียง)
ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์
จุฬาลงกรณ์มหาวิทยาลัย
๑ มกราคม ๒๕๖๕

[1] ชมวิธีใช้งานระบบ Grader ได้ที่ https://youtu.be/Hdnb5rWzGzg

[2] วิดีทัศน์เฉลยอาจยังมีไม่ครบทุกโจทย์



ดูวิดีทัศน์บรรยายเนื้อหา แบบฝึกหัดทบทวน มีตัวตรวจ (ใช้บริการของ **DataCamp**) และเฉลยทุกข้อได้ที่

https://www.cp.eng.chula.ac.th/~somchai/python101

วิชา ๒๑๑๐๑๐๑ ปีการศึกษา ๒๕๕๘ – ๒๕๖๔

คณาจารย์

กุลวดี ศรีพานิชกุลชัย
เจษฎา ธัชแก้วกรพินธ์
ชัยรัต พงศ์พันธุ์ภาณี
เชษฐ พัฒโนทัย
ฐิต ศิริบูรณ์
ดวงดาว วิชาดากุล
ธนารัตน์ ชลิดาพงศ์
ประภาส จงสถิตย์วัฒนา
พรรณราย ศิริเจริญ
พีรพล เวทีกูล
มัณฑนา ปราการสมุทร
วิวัฒน์ วัฒนาวุฒิ
วีระ เหมืองสิน
เศรษฐา ปานงาม
สมชาย ประสิทธิ์จูตระกูล
สืบสกุล พิภพมงคล
สุกรี สินธุภิญโญ
เอกพล ช่วงสุวนิช

อาจารย์ผู้ดูแลระบบ Grader

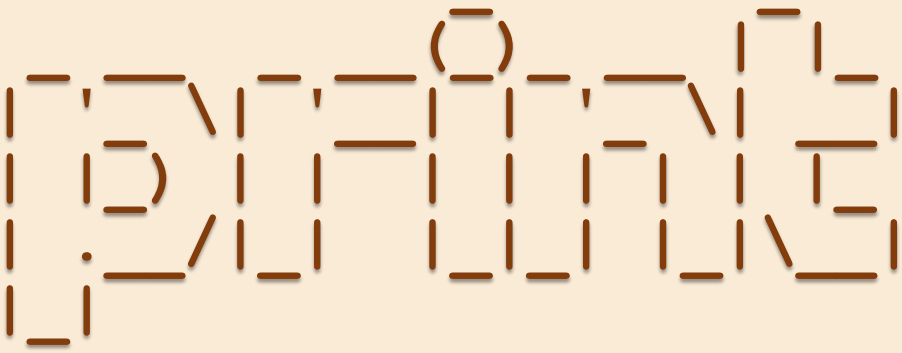
นัทที นิภานันท์

นิสิตช่วยสอน

กนกภัทร จินะณรงค์
กบิล กาญจมาภรณ์กุล
กมลลักษณ์ สุขเสน
กรณัฏฐ์ นฤนาทธนาเสฏฐ์
กรพัฒน์ ปรีชากุล
กฤตย์ กังวานพงศ์พันธุ์
กฤษณะ ขันแก้ว
กวิน มุ่งสมานกูล
กวิน เหลี่ยววงศ์ภูธร
ก้องภพพิสิษฐ์ เติมประทีป
กันย์ แก้วงาม
กิตติภณ พละการ
กิตติภพ พละการ
กิตติศักดิ์ ปรัชญาชูวงศ์
กุลเดช รชตะพฤกษา
ไกรฤกษ์ ตรีทิพสุนทร
ไกรฤกษ์ ตรีทิพสุนทร
ขจรพงษ์ พิมพ์ม่วง
คณิศร ทองประไพแสง
คณุตม์ บุญเรืองขาว
จิรเวช จรูญเวชธรรม
จิรสิญจ์ ชัยมหาวงศ์
เจตนิพิฐ กุลรติวิศว์
เฉลิมวัชช์ พัวพลเทพ
ชยุต ตรีนรินทร์
ชลัช ลิมป์พงศ์สวัสดิ์
ชวิน ช่วงชัยชัชวาล
ชัยวัฒน์ ฉวีววรณ
ชาตรี ชวนงูเหลือม
ชานนท์ ภัทรธิยานนท์
ชุติพนธ์ สาตจีนพงษ์
ณฐวรรธก์ ขวัญภูมิ
ณัฏฐ์ เหลืองสิรพรชัย
ณัฐชนน ผจงกิจพิพัฒน์
ณัฐนัย กิจวิวัฒนการ
ณัฐพงศ์ ศรีวัฒนศักดิ์
ณัฐพจน์ ลิขนะพิชิตกุล
ณัฐภัทร บุญประคอง
ณัษฐพงษ์ อู่สิริมณีชัย
เดชณรงค์ จงจิตสถิตมั่น
เตชิต เผ่าทองบุตร
ธนดล ระงับพิษ
ธนภัทร ลี
ธนัญชัย คงถาวร
ธนากร รัตนจริยา
ธนานพ กอบชัยสวัสดิ์
ธรรมนูญ คุณาพิส
ธวลธรรม จิตรภักดี
ธัชธรรม สังขะเมฆะ
ธีรพงศ์ ปานบุญยืน
ธีรภัทร์ ชินธนกิจ
ธีรภาพ อภิปาลกุล
ธีรวุฒิ พลอาชา
ธีราพร ศุภกุล
นนทเดช วรางค์เดช
บุรินทร์ เนาวรัตน์
ปณิดา ยิ้มสง่า
ปภาณุวัฒน์ พงศ์สวัสดิ์
ปรินทร์ เมปริญญา
ปวิตร เอี่ยมวรวุฒิกุล
ปวินท์ทร ทองสาริ
ปัญจพล สงวนพานิช
พบธรรม วงศ์สมาโนดน์
พลวัต หงส์วิมล
พัชรพล จำรัสพันธุ์
พัทธนันท์ อัครพันธุ์ธัช
พิมพ์รัก อภิรัชตานนท์
พิมพ์วิภา จารุธำรง
พิสิษฐ์ วงศ์ศรีพิสันต์
พิสุทธิ์ จิระรัตนรังษี
เพ็ญภิชา แสงสง่า
ภัทรพร พงษ์ปณตพิพัฒน์
ภูดิท บุญนำเสถียร
มัตตัญญู ตั้งเจ็กกี่
เมวิน มาคารานัส
ยศนัย ชนัศรุติพันธุ์
รวิภาส อภิกุลวณิช
รักษิต รักษาสัตย์
ลัทธพล จีระประดิษฐ
วชิรฉัตร สวัสดิวัตน์ ณ อยุธยา
วรยุทธ วงศ์นิล
วริศ ปุณณะหิตานนท์
วิศวพันธ์ สร้อยเงิน
วุฒิพงศ์ ทาบสุวรรณ
ศิริ ธรรมฤกษ์ฤทธิ์
ศุภกร ชูประภาวรรณ
สรสิช หวังสถิตย์วงษ์
สิทธิพงษ์ เหล่าโก้ก
สุทัศไนย หล้าธรรม
สุปัญญา อภิวงศ์โสภณ
หทัยภัทร สุภานันท์
อธิวรรธน์ ดีโป
อริญจน์ ชวะโนทัย
อริญชย์ ชวะโนทัย
อริยวัฒน์ ชนบดีเฉลิมรุ่ง
อานันท์ เมธเศรษฐ
อิสรา วิชาคำ

สารบัญ

00: Print Statement



00-01: สวัสดีจ้า

จงเขียนโปรแกรมที่แสดงข้อความข้างล่างนี้ทางจอภาพ

```
Hello  Python.
We're using Python 3.
```

ข้อมูลขาเข้า

ไม่มี

ข้อมูลส่งออก

ข้อความที่แสดงข้างบนในโจทย์

ตัวอย่าง

Input (จากแป้นพิมพ์)	**Output** (ทางจอภาพ)
	`Hello  Python.` `We're using Python 3.`

00-02: การหาผลบวกจำนวนเต็มขนาดใหญ่

จงเขียนโปรแกรมที่แสดงผลบวกของ

29384027340912730941623874519287364019263409712345
กับ
92082093849287430982734958720398475092734970

โดยแสดงในรูปแบบที่แสดงในตัวอย่าง

ข้อมูลขาเข้า

ไม่มี

ข้อมูลส่งออก

แสดงตามตัวอย่าง

ตัวอย่าง

Input (จากแป้นพิมพ์)	**Output** (ทางจอภาพ)
	29384027340912730941623874519287364019263409712345 **+** **92082093849287430982734958720398475092734970** **=** *บรรทัดนี้แสดงผลบวกของจำนวนทั้งสองข้างบนนี้*

00-03: ภูเขาและหุบเขา

จงเขียนโปรแกรมที่แสดงรูปข้างล่างนี้

```
+----------------------------------------+
|......................................./\........|
|....................................../##\.......|
|...................................../####\......|
|.................................../######\.....|
|................................../########\....|
|.......__....................../##########\...|
|....../##\.................../############\..|
|./\../####\......../\..../##############\.|
|/##\/######\....../##\../################\|
|############\..../####\/##################|
|############\__/##########################|
+----------------------------------------+
```

การใช้คำสั่ง **`print("\")`** เพื่อต้องการแสดงเครื่องหมาย **backslash `\`** ใน **Python** นั้นทำแบบนี้ไม่ได้ (ลองทำดู จะเกิดปัญหา)

แต่ถ้าใช้คำสั่ง **`print("\\")`** ก็จะทำได้ นั่นคือ ตัว `\` สองตัวติดกันในสตริง จะแทน `\` หนึ่งตัว

(ทำไมถึงเป็นเช่นนี้ จะได้เรียนต่อไป ตอนนี้จำ ๆ ไปก่อน)

ข้อมูลขาเข้า

ไม่มี

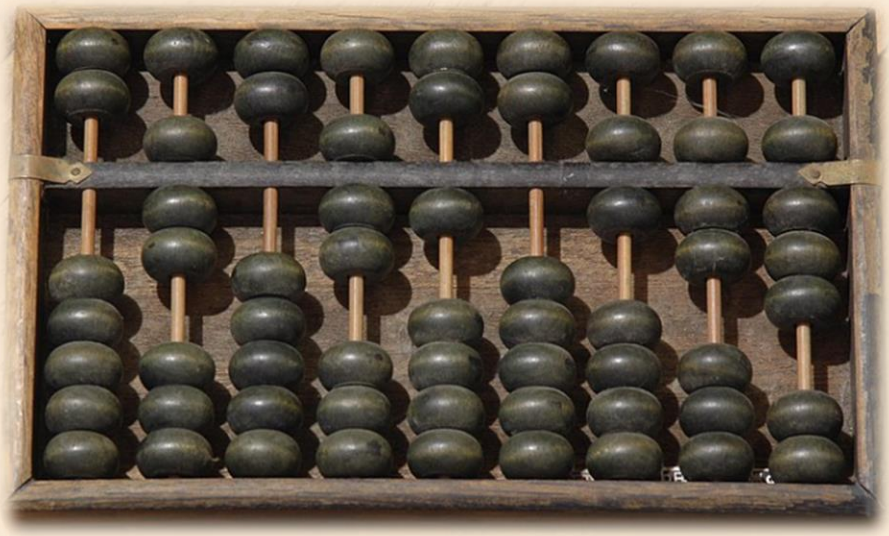
ข้อมูลส่งออก

แสดงตามตัวอย่าง

ตัวอย่าง

Input (จากแป้นพิมพ์)	**Output** (ทางจอภาพ)
	`+----------------------------------------+` `|.....|` `|.....|` `|.....|` `|.....|` `|.....|` `|.....|` `|.....|` `|.....|` `|.....|` `|.....|` `+----------------------------------------+`

01: Data Type and Expression



By ShieldforyoureyesDave Fischer - Own work, CC BY-SA 3.0,
https://commons.wikimedia.org/w/index.php?curid=4954357

01-01: สูตรแปลก ๆ

จงเขียนโปรแกรมที่แสดงผลลัพธ์ของการคำนวณข้างล่างนี้

$$\frac{\pi - \frac{10!}{8^8} + (\log_e 9.7)^{\frac{7}{\sqrt{71}}} - \sin(40^\circ)}{(1.2)^{\sqrt[3]{2.3}}}$$

ข้อมูลนำเข้า

ไม่มี

ข้อมูลส่งออก

แสดงผลลัพธ์ของการคำนวณในโจทย์ (ประมาณ 3.2 กว่า ๆ)

โดยแสดงเลขหลังจุดทศนิยม 6 ตำแหน่ง (ใช้ฟังก์ชัน `round` เช่น `round(2/3, 3)` จะได้ `0.667`)

ตัวอย่าง

ไม่มี

01-02: การประมาณค่าของ n !

ถ้าอยากรู้ว่า 100 ! มีค่าใหญ่ขนาดไหน ก็คงต้องคิดถึง Stirling's approximation ที่คำนวณช่วงของค่า $n!$ ด้วยสูตรข้างล่างนี้

ค่าขอบเขตล่างของ $n!$ ค่าขอบเขตบนของ $n!$

$$\sqrt{2\pi}\, n^{n+\frac{1}{2}} e^{-n+\frac{1}{12n+1}} < n! < \sqrt{2\pi}\, n^{n+\frac{1}{2}} e^{-n+\frac{1}{12n}}$$

จงเขียนโปรแกรมรับจำนวนเต็ม n เพื่อแสดงขอบเขตล่างและบนของการประมาณค่าของ $n!$ จากสูตรข้างบนนี้

ข้อมูลนำเข้า

จำนวนเต็ม n

ข้อมูลส่งออก

ค่าขอบเขตล่าง และค่าขอบเขตบนของ $n!$

ตัวอย่าง

Input (จากแป้นพิมพ์)	**Output** (ทางจอภาพ)
1	0.9958701614627972 1.0022744491822266
5	119.9698539592089 120.00263708619698
50	3.0414009534599554e+64 3.0414093877504934e+64
100	9.332615094728998e+157 9.332621570317666e+157

01-03: สูตรหารากของสมการกำลังสอง

รากจริงของสมการ $ax^2 + bx + c = 0$ คือ

$$x_1 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}, \quad x_2 = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$$

จงเขียนโปรแกรมรับจำนวนจริง a, b และ c เพื่อคำนวณและแสดงรากจริงของสมการ $ax^2 + bx + c = 0$

ข้อมูลนำเข้า

จำนวนจริง a, b และ c บรรทัดละค่า โดยสมการ $ax^2 + bx + c = 0$ ที่ให้มานี้ จะมีรากเป็นค่าจริงสองค่าที่ต่างกันแน่นอน

ข้อมูลส่งออก

รากจริงทั้งสองค่าของสมการ $ax^2 + bx + c = 0$ โดย

- แสดงราก x_1 แล้วตามด้วยราก x_2
- มีเลขหลังจุดทศนิยม 3 ตำแหน่ง (ใช้ฟังก์ชัน `round` เช่น `round(2/3, 3)` จะได้ `0.667`)

ตัวอย่าง

Input (จากแป้นพิมพ์)	**Output** (ทางจอภาพ)
`1.0` `-5.0` `6.0`	`2.0 3.0`
`1.0` `-1` `-42`	`-6.0 7.0`
`6` `-4.0` `-12`	`-1.12 1.786`
`20.0` `-50.5` `-21.2`	`-0.367 2.892`

01-04: พื้นที่ผิวกาย

พื้นที่ผิวกาย (body surface area) เป็นค่าหนึ่งที่มักในวงการแพทย์เพื่อกำหนดปริมาณยาที่ใช้ในการรักษา มีสูตรในการประมาณพื้นที่ผิวกายหลายสูตรดังแสดงข้างล่างนี้ (W คือน้ำหนัก หน่วยเป็นกิโลกรัม H คือความสูง หน่วยเป็นเซนติเมตร)

สูตรของ Mosteller	$\frac{\sqrt{W \times H}}{60}$
สูตรของ Haycock	$0.024265 \times W^{0.5378} \times H^{0.3964}$
สูตรของ Boyd	$0.0333 \times W^{(0.6157 - 0.0188 \log_{10} W)} \times H^{0.3}$

จงเขียนโปรแกรมที่รับค่าน้ำหนักและส่วนสูง แล้วแสดงค่าพื้นที่ผิวกายที่คำนวณได้จากสูตรทั้งสามข้างบนนี้

ข้อมูลนำเข้า

บรรทัดแรกเป็นจำนวนจริงแทนน้ำหนักหน่วยเป็นกิโลกรัม

บรรทัดที่สองเป็นจำนวนจริงแทนความสูงหน่วยเป็นเซนติเมตร

ข้อมูลส่งออก

ค่าพื้นที่ผิวกายที่คำนวณได้จากสูตรของ Mosteller, Haycock และ Boyd บรรทัดละค่า

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
56 173	1.6404606399152375 1.6304868174022364 1.632155747802396
60 170	1.6832508230603465 1.680428314258862 1.6863370568707923
80.0 150.0	1.8257418583505538 1.8666576124395382 1.9007070607658065

01-05: ช่วงเวลา

โปรแกรมทางขวานี้รับเลขชั่วโมง นาที และวินาที ของเวลาเริ่มกับเวลาสิ้นสุด จากนั้นแสดงช่วงเวลาเป็นจำนวนชั่วโมง นาที และวินาที ระหว่างเวลาเริ่มถึงสิ้นสุด โดยมีข้อจำกัดว่า เวลาสิ้นสุดจะต้องไม่น้อยกว่าเวลาเริ่มต้น

เช่น ถ้าป้อนเลข **2 10 20 4 0 0** บรรทัดละจำนวน จะได้ผลลัพธ์คือ **1:49:40** แต่ถ้าป้อน **2 0 0 1 0 0** บรรทัดละจำนวน จะได้ผลลัพธ์คือ **-1:0:0** ซึ่งผิด ที่ถูกควรเป็น **23:0:0**

```
h1 = int(input())
m1 = int(input())
s1 = int(input())
h2 = int(input())
m2 = int(input())
s2 = int(input())
t1 = h1*60*60 + m1*60 + s1
t2 = h2*60*60 + m2*60 + s2
dt = t2 - t1
dh = dt // (60*60)
dt -= dh * 60*60
dm = dt // 60
dt -= dm*60
ds = dt
print(str(dh) + ":" + \
      str(dm) + ":" + str(ds))
```

จงปรับปรุงโปรแกรมข้างต้นให้ถูกต้องทั้งในกรณีที่รับเวลาสิ้นสุดมากกว่า น้อยกว่าหรือเท่ากับ เวลาเริ่มต้น (กำหนดให้ช่วงเวลาไม่เกิน 23:59:59)

ข้อแนะนำ : ถ้าเราสนใจเฉพาะเลขชั่วโมง การคำนวณช่วงเวลาจาก h1 ถึง h2

- แบบง่าย ๆ ก็เท่ากับ h2 - h1 เช่น h1 = 1 ถึง h2 = 2 ก็เท่ากับ h2 - h1 = 2 - 1 = 1 ชั่วโมง ซึ่งจะใช้ได้ก็เมื่อ h2 ≥ h1
- ถ้าสลับกัน ให้ h1 = 2 และ h2 = 1 ช่วงเวลา 2 นาฬิกา ถึง 1 นาฬิกา ย่อมไม่เท่ากับ 1 - 2 = -1 แต่เท่ากับ 23 ชั่วโมง ถ้าดูดี ๆ 23 = 24 + (-1) จึงขอแก้สูตรช่วงเวลาจาก h1 ถึง h2 ให้เท่ากับ 24 + (h2 - h1) ก็จะใช้ได้ในกรณี h2 < h1
- ถ้าปรับสูตรให้เป็น (24 + (h2 - h1)) % 24 ก็สามารถใช้ได้ไม่ว่า h2 ≥ h1 หรือ h2 < h1 (ลองดูเอง)
- (หรือใช้สูตรแค่ (h2 - h1) % 24 ก็ใช้ได้เหมือนกัน จะเข้าใจตรงนี้ ต้องเข้าใจการใช้ % กับจำนวนลบ ซึ่งไม่ขออธิบาย)

ข้อมูลนำเข้า

สามบรรทัดแรกรับ เลขชั่วโมง นาที และวินาที ของเวลาเริ่มต้น บรรทัดละจำนวน
ตามด้วยอีกสามบรรทัดที่รับ เลขชั่วโมง นาที และวินาที ของเวลาสิ้นสุด บรรทัดละจำนวน
(ชั่วโมงเป็นจำนวนเต็ม 0 ถึง 23 ส่วนนาทีและวินาทีเป็นจำนวนเต็ม 0 ถึง 59)

ข้อมูลส่งออก

ช่วงเวลาตั้งแต่เวลาเริ่มจนสิ้นสุด (ที่รับเข้ามา) แสดงเป็นจำนวนชั่วโมง นาที และวินาที ในรูปแบบที่แสดงในตัวอย่าง

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
2 10 20 4 0 0	1:49:40
18 10 10 19 0 0	0:49:50
19 0 0 18 10 10	23:10:10

02: Basic String & List

By FranHogan - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=92636750

02-01: เลขประจำตัวประชาชน

เลขบัตรประจำตัวประชาชนของคนไทยประกอบด้วยเลข 13 หลัก กำหนดให้ n_0 คือเลขตัวซ้ายสุด ไล่ไปจนถึง n_{12} คือเลขตัวขวาสุด เลขตัวขวาสุดนี้มีค่าที่คำนวณได้จากเลข 12 ตัวทางซ้าย มีไว้เพื่อตรวจสอบว่า มีการป้อนเลขบัตรผิดหรือไม่ (ซึ่งตรวจได้ระดับหนึ่ง) ในวงการเรียกเลขนี้ว่า check digit มีสูตรการคำนวณดังนี้

$$n_{12} = (11 - (13n_0+12n_1+11n_2+10n_3+9n_4+8n_5+7n_6+6n_7+5n_8+4n_9+3n_{10}+2n_{11}) \bmod 11) \bmod 10$$

จงเขียนโปรแกรมเพื่อหา check digit ของ เลข 12 หลักแรกของเลขบัตรประจำตัวประชาชน และแสดงเลขบัตรตามรูปแบบมาตรฐาน

ข้อมูลนำเข้า

เลข 12 หลักแรก (จากซ้าย) ของเลขที่บัตรประชาชน

ข้อมูลส่งออก

เลข 12 หลักที่รับมา พร้อมกับ เลข check digit ในรูปแบบมาตรฐานที่ปรากฏในบัตรประชาชน (ดูตัวอย่างประกอบ)

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
`123456789012`	`1 2345 67890 12 1`
`310030011214`	`3 1003 00112 14 2`
`110070234512`	`1 1007 02345 12 9`

02-02: เลขอารบิก

จงเขียนโปรแกรมที่อ่านเลข 1 ตัว แล้วแสดงคำอ่านในภาษาอังกฤษ ตามตารางข้างล่างนี้

0	1	2	3	4	5	6	7	8	9
zero	one	two	three	four	five	six	seven	eight	nine

ข้อมูลนำเข้า

เลข 1 ตัว

ข้อมูลส่งออก

คำอ่านเลขที่ได้รับในภาษาอังกฤษ

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
`5`	`5 --> five`
`1`	`1 --> one`

02-03: วันที่

จงเขียนโปรแกรมที่อ่านวันเดือนปีในรูปแบบ เลขวัน/เลขเดือน/เลขปี เพื่อเปลี่ยนและแสดงในรูปแบบ ชื่อเดือน เลขวัน, เลขปี

ข้อมูลนำเข้า

วันที่ในรูปแบบ เลขวัน/เลขเดือน/เลขปี

ข้อมูลส่งออก

วันที่ในรูปแบบ ชื่อเดือน เลขวัน, เลขปี

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
`12/1/2019`	`January 12, 2019`
`31/12/2020`	`December 31, 2020`

02-04: นำหน้าด้วย 0

จงเขียนโปรแกรมรับจำนวนเต็มบวก **M** กับ **N** เพื่อแสดงค่า **M** ทางจอภาพ โดยถ้าค่า **M** มีจำนวนหลักน้อยกว่า **N** ก็ให้เติม 0 ด้านซ้าย ให้จำนวนหลักทั้งหมดครบ **N** ตัว เช่น ถ้า **M=123**, **N=5** จะแสดง **00123** ถ้า **M=12345**, **N=3** จะแสดง **12345**

ข้อมูลนำเข้า

จำนวนเต็มบวก 2 จำนวน สำหรับ **M** กับ **N** จำนวนละบรรทัด

ข้อมูลส่งออก

ค่าของ **M** ที่อาจมีการเติม 0 ด้านซ้ายเพื่อให้มีจำนวนหลักอย่างน้อย **N** หลัก

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
123 5	00123
123 3	123
123 2	123

ข้อแนะนำ: Python มี built-in function ชื่อ **max** มีไว้ใช้หาค่ามากสุดของข้อมูล เช่น **max(5,9)** ได้ **9**

02-05: ยอดขายทั้งสัปดาห์

จงเขียนโปรแกรมรับยอดขายของแต่ละวันในหนึ่งสัปดาห์ เพื่อหาและแสดงยอดขายรวมของทั้งสัปดาห์

ข้อมูลนำเข้า

จำนวนเต็ม 7 ตัว เรียงกันในบรรทัดเดียว คั่นด้วยช่องว่าง

ข้อมูลส่งออก

แสดงผลรวมของจำนวนทั้ง 7 ที่รับเข้ามา

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
`10 20 30 40 50 60 70`	`280`
`0 0 0 100 0 0 0`	`100`

02-06: ผลบวกเวกเตอร์ 3 มิติ

ให้ $\mathbf{u} = [u_1, u_2, u_3]$ และ $\mathbf{v} = [v_1, v_2, v_3]$ เป็นเวกเตอร์ ผลบวกของเวกเตอร์ทั้งสองมีค่าเท่ากับ

$$\mathbf{u} + \mathbf{v} = [u_1 + v_1,\ u_2 + v_2,\ u_3 + v_3]$$

จงเขียนโปรแกรมรับเวกเตอร์สามมิติสองตัว แล้วแสดงผลบวกของเวกเตอร์ทั้งสอง

ข้อมูลนำเข้า

เวกเตอร์สามมิติสองตัว บรรทัดละตัว ในรูปแบบ **[***จำนวน***,** *จำนวน***,** *จำนวน***]**

ข้อมูลส่งออก

ผลบวกเวกเตอร์ในรูปแบบเดียวกับที่รับ (ดูตัวอย่างข้างล่าง)

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
`[1, 2, 3]` `[2, 3, 4]`	`[1.0, 2.0, 3.0] + [2.0, 3.0, 4.0] = [3.0, 5.0, 7.0]`
`[1.25, 2.5, 3.0]` `[-2.5, 1.3, -2.5]`	`[1.25, 2.5, 3.0] + [-2.5, 1.3, -2.5] = [-1.25, 3.8, 0.5]`
`[0, 10, 0]` `[10, 0, 10]`	`[0.0, 10.0, 0.0] + [10.0, 0.0, 10.0] = [10.0, 10.0, 10.0]`

ข้อแนะนำ: ถ้า `d = [1.2, 3.4, 5.6]` เป็นลิสต์ คำสั่ง `print(d)` จะได้แสดง `[1.2, 3.4, 5.6]` ทางจอภาพ

02-07: ถอดรหัสลับ

นาย ก ต้องการส่งรหัสลับให้นาย ข โดยส่งเป็นเลข 0 ถึง 9 จำนวน 32 ตัว แล้วตกลงกับนาย ข ว่า วิธีถอดรหัสลับเป็นดังนี้ (ดูตัวอย่างประกอบด้วย)

กำหนดให้ เลขซ้ายสุดคือหลักที่ 1

1. หยิบเลขหลักที่ 4,11,18,25 และ 32 (คือเริ่มที่ 4 แล้วข้ามไปทีละ 7) มาเขียนติดกัน
2. หยิบเลขหลักที่ 8,13,18,23,28 (คือเริ่มที่ 8 แล้วข้ามไปทีละ 5) มาเขียนติดกัน
3. นำเลขทั้งสองที่ได้จาก 2 ข้อแรกมาบวกกันแล้วบวกอีก 10000
4. เลือกเฉพาะหลักพัน หลักร้อย และหลักสิบของจำนวนที่ได้ในข้อ 3
5. นำแต่ละหลักของจำนวนในข้อ 4 มารวมกัน เลือกหลักหน่วย แล้วเพิ่มอีก 1
6. แปลงเลขที่ได้ในข้อที่ 5 เป็นตัวอังกฤษตัวใหญ่ โดย 1 คือ A, 2 คือ B, 3 คือ C, ..., 9 คือ I, 10 คือ J
7. รหัสลับที่ต้องการส่งคือ เลขจากข้อที่ 4 ต่อด้วยตัวอักษรจากข้อที่ 6

```
12345678901234567890123456789012  <-- หลักที่
92813912398100282033745980018127  <-- ข้อมูล

92813912398100282033745980018127 ได้ 18087

92813912398100282033745980018127 ได้ 20051

18087 + 20051 + 10000 = 48138

48138 --> 813

813 --> 8 + 1 + 3 = 12 --> 2 + 1 = 3

3 --> C

813C
```

ข้อมูลนำเข้า

ตัวเลขจำนวน 32 หลัก

ข้อมูลส่งออก

รหัสที่ถอดได้จากข้อมูลนำเข้าด้วยวิธีที่กำหนดให้ข้างต้น

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
92813912398100282033745980018127	813C
00000000000000000000000000000000	000A
99999999999999999999999999999999	999H

02-08: จำนวนทศนิยมและเศษส่วน

เราสามารถเขียนจำนวนตรรกยะในรูปแบบเศษส่วนหรือแบบทศนิยมได้ เช่น $\frac{1}{8} = 0.125$ แต่ก็มีจำนวนตรรกยะที่เขียนออกมาได้เป็นเลขหลังจุดทศนิยมไม่รู้จบแบบซ้ำ เช่น $\frac{3221}{555} = 5.8036036036036036...$ (เลข 036 จะซ้ำไปเรื่อย ๆ ไม่รู้จบ) ในกรณีนี้ ขอเขียนเป็น 5.8(036) แสดงให้เห็นว่า เลขในวงเล็บ 036 จะซ้ำไม่รู้จบ จงเขียนโปรแกรมที่รับจำนวนในรูปแบบทศนิยม แล้วแสดงในรูปแบบเศษส่วน

ข้อมูลนำเข้า

จำนวนไม่ติดลบแบบทศนิยม ที่แบ่งทศนิยมเป็นสามส่วนคั่นด้วยจุลภาคคือ เลขหน้าจุด เลขหลังจุดที่ไม่อยู่ในวงเล็บ และเลขในวงเล็บ (ดูตัวอย่าง)

ข้อมูลส่งออก

จำนวนในรูปแบบเศษส่วนที่มีค่าเดียวกับจำนวนที่รับเข้ามา โดยที่ค่าของเศษและส่วนมี ห.ร.ม. เป็น 1 (ดูตัวอย่าง)

ตัวอย่าง

จำนวนในรูปแบบทศนิยม	input (จากแป้นพิมพ์)	output (ทางจอภาพ)
7.	`7,,0`	`7 / 1`
0.	`0,,0`	`0 / 1`
0.5	`0,5,0`	`1 / 2`
0.08(3)	`0,08,3`	`1 / 12`
0.02(27)	`0,02,27`	`1 / 44`
123.456(789)	`123,456,789`	`41111111 / 333000`
987.(987)	`987,,987`	`329000 / 333`

พยายามเขียน **code** โดยใช้เฉพาะคำสั่งในบทที่ 2 (คือไม่ใช้คำสั่ง **if ...**)

ข้อแนะนำ

เราสามารถใช้บริการ **math.gcd(a,b)** ในการหา ห.ร.ม ของ **a** กับ **b** เช่น คำสั่ง **math.gcd(2431, 13277)** ได้ผลเป็น **187** ดังนั้น

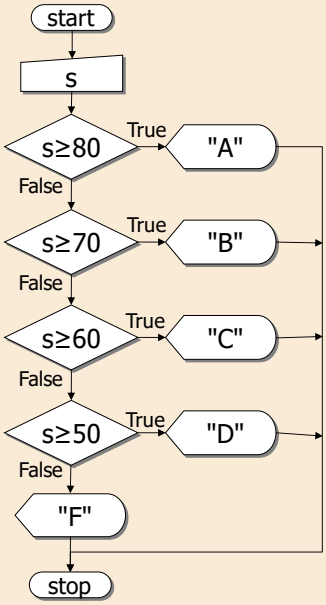
$$\frac{2431}{13277} = \frac{2431/187}{13277/187} = \frac{13}{71}$$

03: Selection: if-elif-else

By CrisNYCa - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=94532194

03-01: ผังงานการตัดเกรด

จงเขียนโปรแกรมแสดงเกรดที่ได้จากคะแนนที่รับมา โดยมีเกณฑ์การตัดเกรดตามที่แสดงในผังงานทางขวานี้



ข้อมูลนำเข้า

จำนวนจริงหนึ่งจำนวน แทนคะแนน

ข้อมูลส่งออก

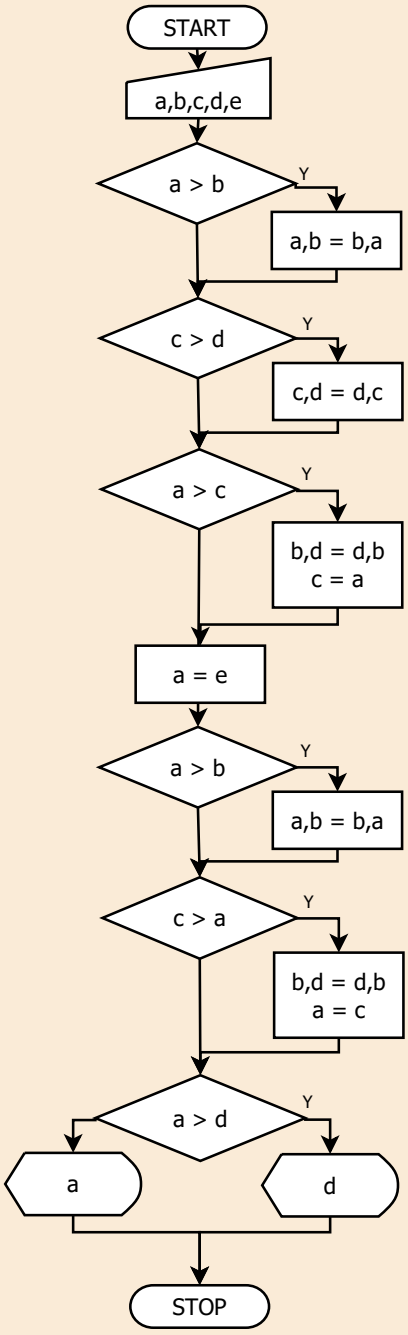
เกรดที่ได้รับตามเกณฑ์การตัดเกรดทางขวา

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
`99.99`	`A`
`0.1`	`F`

03-02: มัธยฐานของห้าจำนวน

จงเขียนโปรแกรมที่หามัธยฐานของข้อมูล 5 ตัว ที่ทำงานตาม flowchart ข้างขวานี้



ข้อมูลนำเข้า

จำนวนเต็ม 5 จำนวน บรรทัดละจำนวน

ข้อมูลส่งออก

แสดงมัธยฐานของข้อมูลที่รับเข้ามา

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
1 2 3 4 5	3
40 30 20 50 10	30
10 10 10 10 20	10
0 -1 -2 2 1	0

03-03: สิบห้าวันถัดไป

จงเขียนโปรแกรมที่หาคำตอบว่า ถัดจากวันเดือนปีที่กำหนดให้ ไปอีก 15 วันคือวันเดือนปีอะไร

หมายเหตุ: ถ้า เลขปี ค.ศ. หารด้วย 400 ลงตัว หรือ หารด้วย 4 ลงตัว แต่หารด้วย 100 ไม่ลงตัว เดือนกุมภาพันธ์ในปีนั้นก็มี 29 วัน

ข้อมูลนำเข้า

จำนวนเต็ม 3 จำนวน คือ เลขวัน เลขเดือน และเลขปี พ.ศ.

หมายเหตุ: การอ่าน input นี้ใช้คำสั่ง

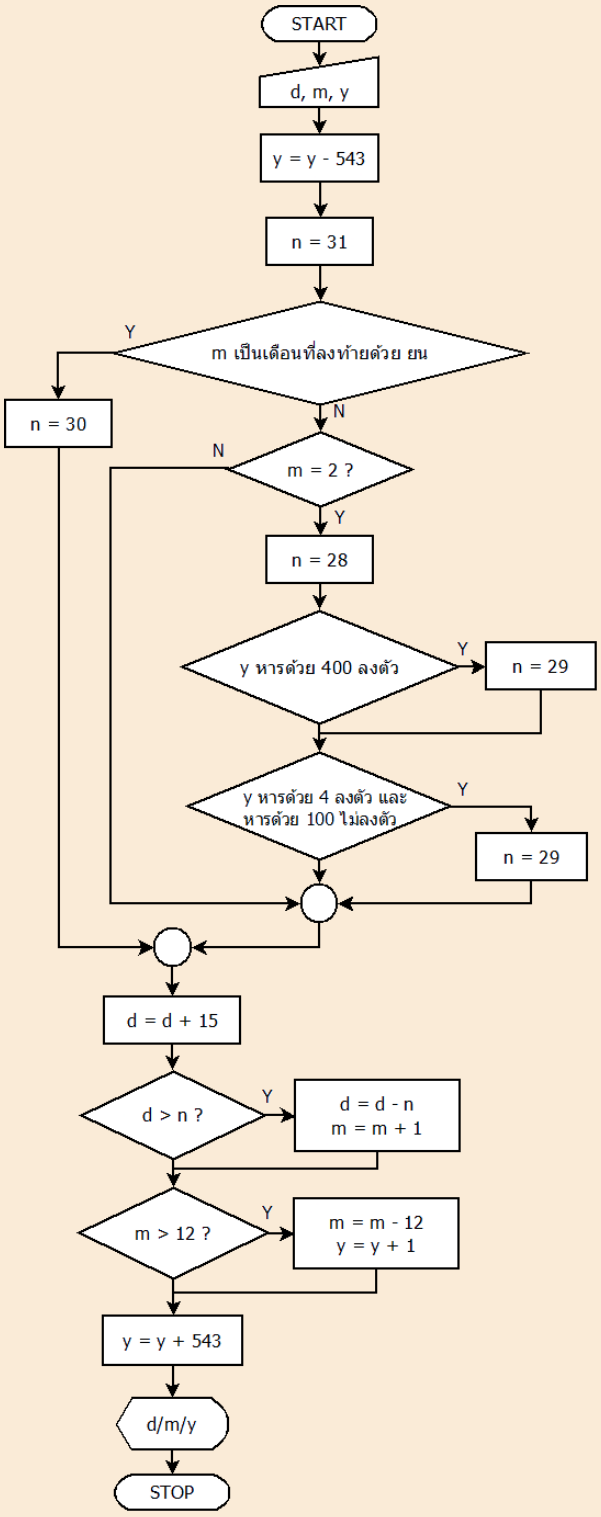
```
d,m,y = [int(e) for e in input().split()]
```

ข้อมูลส่งออก

เลขวันเดือนปีที่ถัดจากวันที่ได้รับอีก 15 วัน ในรูปแบบดังแสดงในตัวอย่าง

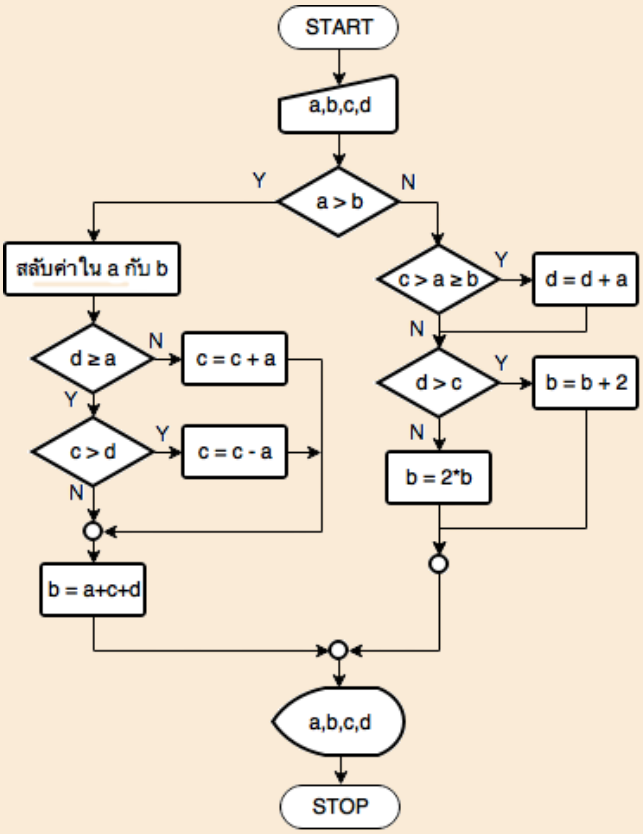
ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
`1 1 2560`	`16/1/2560`
`31 12 2560`	`15/1/2561`
`28 2 2559`	`14/3/2559`
`28 2 2560`	`15/3/2560`



03-04: ผังงานการตัดสินใจ

จงเขียนโปรแกรมที่ทำงานตามผังงานทางขวานี้



ข้อมูลนำเข้า

จำนวนเต็ม 4 จำนวนบนบรรทัดเดียวกัน คั่นด้วยช่องว่าง

ข้อมูลส่งออก

ค่าของ a b c และ d ในผังงาน คั่นด้วยช่องว่างหนึ่งช่อง

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
`1 2 3 4`	`1 4 3 4`
`4 3 2 1`	`3 9 5 1`
`2 2 2 2`	`2 4 2 2`
`9 0 9 0`	`0 9 9 0`
`3 2 1 4`	`2 7 1 4`

03-05: รหัสคณะ

เว็บไซต์สำนักทะเบียนจุฬาฯ กำหนดรหัสให้กับคณะ/สถาบันต่าง ๆ ดังแสดงในตารางข้างล่างนี้

01	สถาบันภาษาไทยสิรินธร	32	คณะทันตแพทยศาสตร์
02	ศูนย์การศึกษาทั่วไป	33	คณะเภสัชศาสตร์
20	บัณฑิตวิทยาลัย	34	คณะนิติศาสตร์
21	คณะวิศวกรรมศาสตร์	35	คณะศิลปกรรมศาสตร์
22	คณะอักษรศาสตร์	36	คณะพยาบาลศาสตร์
23	คณะวิทยาศาสตร์	37	คณะสหเวชศาสตร์
24	คณะรัฐศาสตร์	38	คณะจิตวิทยา
25	คณะสถาปัตยกรรมศาสตร์	39	คณะวิทยาศาสตร์การกีฬา
26	คณะพาณิชยศาสตร์และการบัญชี	40	สำนักวิชาทรัพยากรการเกษตร
27	คณะครุศาสตร์	51	วิทยาลัยประชากรศาสตร์
28	คณะนิเทศศาสตร์	53	วิทยาลัยวิทยาศาสตร์สาธารณสุข
29	คณะเศรษฐศาสตร์	55	สถาบันภาษา
30	คณะแพทยศาสตร์	58	สถาบันบัณฑิตบริหารธุรกิจศศินทร์ฯ
31	คณะสัตวแพทยศาสตร์		

จงเขียนโปรแกรมตัวเลขมาตรวจสอบว่าเป็นรหัสคณะ/สถาบันที่ถูกต้องหรือไม่

ข้อมูลนำเข้า

ลำดับอักขระจำนวนหนึ่ง

ข้อมูลส่งออก

ข้อความ **OK หรือ Error เพื่อระบุว่าลำดับอักขระรับว่าเป็นรหัสคณะฝสถาบันที่ถูกต้องตามตารางข้างต้นหรือไม่**

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
`Engineering`	`Error`
`1`	`Error`
`10`	`Error`
`20000`	`Error`
`58`	`OK`
`01`	`OK`

03-06: การย้ายภาค

สมมติว่าภาควิชาวิศวกรรมคอมพิวเตอร์ ตั้งเกณฑ์การย้ายภาคดังนี้ (ขอเน้นว่านี่เป็นเรื่องสมมติ ปัจจุบันไม่ได้ใช้เกณฑ์ข้างล่างนี้)

1. ต้องได้เกรด comp prog เป็น A และเกรดของ cal1 กับ cal2 อย่างน้อย C
2. ถ้าต้องเลือกนิสิตคนเดียวระหว่างนิสิตสองคนที่ผ่านเกณฑ์ข้อที่ 1 จะเลือกพิจารณาดังนี้
   - เลือกคนที่มี GPAX มากกว่า
   - ถ้า GPAX เท่ากัน เลือกคนที่ได้เกรด cal1 สูงกว่า
   - ถ้า GPAX เท่ากัน และเกรด cal1 ก็เท่ากัน ก็เลือกคนที่ได้เกรด cal2 สูงกว่า
   - ถ้า GPAX, cal1 และ cal2 เท่ากันหมด ก็จำเป็นต้องเลือกทั้งคู่

จงเขียนโปรแกรมรับข้อมูลนิสิตสองคน แล้วแสดงผลว่าจะเลือกใครดี

ข้อมูลนำเข้า

สองบรรทัด แต่ละบรรทัดประกอบด้วย เลขประจำตัวนิสิต GPAX เกรดของ comp prog เกรดของ cal1 และ เกรดของ cal2

(GPAX เป็นจำนวนจริง และ เกรดเป็นตัวอักษร A, B, C, D หรือ F แน่ ๆ ไม่มีประจุ)

ข้อมูลส่งออก

ถ้าทั้งคู่ไม่ผ่านเกณฑ์ข้อที่ 1 แสดง **None**

ถ้าผ่านเกณฑ์ข้อที่ 1 คนเดียว แสดงเลขประจำตัวของคนนั้น

ถ้าผ่านเกณฑ์ข้อที่ 2 ทั้งคู่ แสดงเลขประจำตัวของคนที่มีถูกเลือกด้วยเกณฑ์ข้อที่ 2

ถ้าทั้งคู่มีเกณฑ์ข้อที่ 2 เท่ากัน แสดง **Both**

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
7039999921 2.8 B C C 7030000021 3.5 B A A	None
7039999921 2.8 A C C 7030000021 3.5 B A A	7039999921
7030000021 3.2 A A D 7039999921 2.8 A C C	7039999921
7039999921 3.1 A B B 7030000021 3.0 A A A	7039999921
7039999921 3.1 A B B 7030000021 3.1 A C A	7039999921
7039999921 3.1 A C A 7030000021 3.1 A C C	7039999921
7039999921 3.1 A C B 7030000021 3.1 A C B	Both

03-07: คะแนนยิมนาสติก

การให้คะแนนในกีฬายิมนาสติกแบบหนึ่ง จะพิจารณาจากคะแนนของกรรมการ 4 คน ตัดคะแนนที่ต่ำสุดและสูงสุดออก แล้วนำคะแนนของสองคนที่เหลือมาหาค่าเฉลี่ย

จงเขียนโปรแกรมที่รับคะแนนจากกรรมการ 4 คน แล้วแสดงคะแนนสุดท้ายที่คำนวณได้ด้วยวิธีข้างต้น

ข้อมูลนำเข้า

จำนวนจริง 4 จำนวนบนบรรทัดเดียวกัน คั่นด้วยช่องว่าง

ให้ฝึกใช้คำสั่ง **`if`** เพื่อหาค่ามากสุดและน้อยสุด (ควรหลีกเลี่ยงการใช้ **`min`** กับ **`max`**)

ข้อมูลส่งออก

คะแนนสุดท้ายที่คำนวณได้ (แสดงเลขจุดทศนิยม 2 ตำแหน่ง)

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
`9.84 9.30 9.42 9.58`	`9.5`
`9.15 9.20 9.30 9.50`	`9.25`

03-08: หมายเลขโทรศัพท์เคลื่อนที่

หมายเลขโทรศัพท์จะเป็นหมายเลขโทรศัพท์เคลื่อนที่ ก็เมื่อเป็นตัวเลข 10 ตัว และขึ้นต้นด้วยเลข 06, 08 หรือ 09

จงเขียนโปรแกรมรับตัวเลขติดกันจำนวนหนึ่ง จากนั้นแสดงให้เห็นว่าเป็นเลขที่ได้รับเป็นหมายเลขโทรศัพท์เคลื่อนที่หรือไม่เป็น

ข้อมูลนำเข้า

ตัวเลขติดกันจำนวนหนึ่ง (ข้อมูลที่ได้รับเป็นตัวเลขหมดทุกหลักแน่ ๆ)

ข้อมูลส่งออก

ข้อความ **Mobile number หรือ Not a mobile number ขึ้นกับว่าข้อมูลที่ได้รับเป็นหมายเลขโทรศัพท์เคลื่อนที่หรือไม่**

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
`1112`	`Not a mobile number`
`022153555`	`Not a mobile number`
`0868144444`	`Mobile number`
`0900999999`	`Mobile number`
`0646664444`	`Mobile number`
`0700100010`	`Not a mobile number`

03-09: บวก-ลบ-ศูนย์-คู่-คี่ 

จำนวนเต็มสามารถจำแนกประเภทได้หลายแบบ เช่น จำนวนบวก/ศูนย์/ลบ หรือ จำนวนคู่/คี่

โจทย์ข้อนี้ให้ทดลองใช้คำสั่งแบบเลือกทำเพื่อจำแนกประเภทของจำนวนที่กำหนด

ข้อมูลนำเข้า

บรรทัดเดียว เป็นจำนวนเต็ม

ข้อมูลส่งออก

สองบรรทัด บรรทัดแรกระบุว่าจำนวนที่รับมาเป็นจำนวนบวก (**`positive`**) ศูนย์ (**`zero`**) หรือจำนวนลบ (**`negative`**) อีกบรรทัดระบุว่าเป็นจำนวนคู่ (**`even`**) หรือจำนวนคี่ (**`odd`**)

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
`12`	`positive` `even`
`-35`	`negative` `odd`
`0`	`zero` `even`

03-10: ค่าส่งพัสดุลงทะเบียน

จงเขียนโปรแกรมรับน้ำหนักพัสดุ (หน่วยเป็นกรัม) เพื่อแสดงค่าส่งพัสดุลงทะเบียนตามเกณฑ์ที่แสดงข้างล่างนี้

http://cp.lnwfile.com/_/cp/_raw/uf/kp/c2.jpg

ข้อมูลนำเข้า

จำนวนเต็มหนึ่งจำนวน แทนน้ำหนักพัสดุ (หน่วยเป็นกรัม)

ข้อมูลส่งออก

ค่าส่งพัสดุลงทะเบียน ตามตารางข้างบน

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
50	18
300	28
5000	Reject

03-11: การแสดงตัวเลขแบบย่อ

ก่อนเดือนพฤษภาคม 2019 YouTube แสดงจำนวน subscribers ว่ามีกี่คนอย่างละเอียด เช่น 1,454,039 แต่ได้ปรับวิธีแสดงแบบย่อ เช่น 1.5M ซึ่งก็คือสิ่งที่ต้องเขียนเป็นโปรแกรมในงานนี้

ข้อมูลนำเข้า

จำนวนเต็มหนึ่งจำนวน

ข้อมูลส่งออก

รูปแบบการแสดงจำนวนแบบย่อ (ขอให้พิจารณากรณีต่าง ๆ จากตัวอย่างข้างล่างนี้)

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
32	32
8456	8.5K
84560	85K
108283	108K
2293910	2.3M
12999995	13M
580912391	581M
1301008191	1.3B
25555555555	26B
2555555555555	2556B

03-12: วันที่เท่าไรของปี

จงเขียนโปรแกรมที่รับเลขวัน เลขเดือน และเลขปี (พ.ศ.) เพื่อแสดงว่าเป็นวันที่เท่าไรของปี

ข้อมูลนำเข้า

บรรทัดแรกคือ เลขวัน

บรรทัดที่สองคือ เลขเดือน

บรรทัดที่สามคือ เลขปี (พ.ศ.)

ข้อมูลส่งออก

เลขวันที่ของปี

ตัวอย่าง

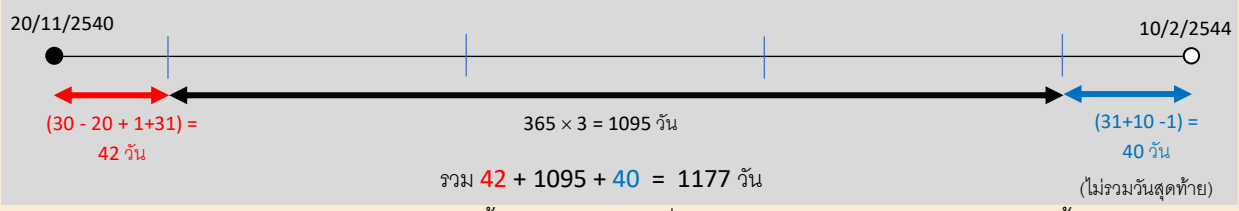
input (จากแป้นพิมพ์)	output (ทางจอภาพ)
1 1 2559	1
31 12 2559	366
5 12 2560	339
31 12 2560	365

03-13: จังหวะชีวิต

Wilhelm Fliess นักวิทยาศาสตร์ชาวเยอรมันได้เสนอทฤษฎี Biorhythm ที่ระบุว่า ความสามารถด้านต่าง ๆ ของมนุษย์มีขึ้นมีลงเป็นรอบ ๆ ตามจำนวนวันตั้งแต่เกิดมา ที่ใช้กันมากก็คือ ความสามารถด้าน physical, emotional และ intellectual จะขึ้นลงเป็น 23, 28 และ 33 วันต่อรอบตามลำดับ มีสูตรดังนี้ ทาง physical คือ $\sin\left(\frac{2\pi t}{23}\right)$, ทาง emotional คือ $\sin\left(\frac{2\pi t}{28}\right)$ และทาง intellectual คือ $\sin\left(\frac{2\pi t}{33}\right)$ ซึ่งมีค่าอยู่ระหว่าง -1 ถึง 1 โดยที่ t คือจำนวนวันตั้งแต่ได้เกิดมา (ขอไม่อธิบายความหมายของค่า biorhythm ต่าง ๆ ผู้สนใจหาอ่านในเน็ตเอง)

โจทย์ข้อนี้รับ วันเดือนปีเกิด และวันเดือนปีที่สนใจรู้ค่า biorhythm ทั้งสาม เพื่อคำนวณและแสดงค่าทั้งสาม ก่อนอื่นต้องคำนวณจำนวนวันตั้งแต่เกิด หาได้โดยรวมจำนวนวันของสามช่วง (ดูรูปข้างล่างนี้) คือ ช่วงแดงจากวันเกิดจนถึงปลายปีเกิด ช่วงดำ และ ช่วงฟ้าตั้งแต่ต้นปีที่สนใจถึงวันก่อนวันที่สนใจ (คือไม่รวมวันที่สนใจค่า biorhythm) และเพื่อให้การคำนวณง่ายขึ้น ขอกำหนดให้

- การหาช่วงดำ ทำโดยนำจำนวนปีคูณด้วย 365 เช่น 3 ปี = 365×3 = 1095 วัน แต่สำหรับช่วงแดงและฟ้า ต้องคำนวณจำนวนวันให้ตรงเป๊ะ (ซึ่งคงต้องใช้หลาย ๆ if และคำนึงถึงกรณีเดือนกุมภามี 28 หรือ 29 วันด้วย)
- วันเดือนปีทั้งสองที่รับมา จะอยู่คนละปี (ทำให้มั่นใจว่ามีช่วงแดง และฟ้าแน่นอน แต่ช่วงดำอาจเป็น 0)



หมายเหตุ : การคำนวณจำนวนวันในช่วงดำข้างบนนี้จะได้ค่าประมาณที่อาจผิดพลาดหลายวัน แต่นิสิตต้องใช้วิธีนี้ในโปรแกรม (หากใช้วิธีอื่นที่เที่ยงตรงกว่า ก็อาจได้ผลไม่ตรงตามตัวตรวจใน Grader)

ข้อมูลนำเข้า

จำนวนเต็ม 6 จำนวนแทนเลขวัน เดือน ปี (พ.ศ.) ของวันเกิด ตามด้วยเลขวัน เดือน ปีที่สนใจหาค่าของ biorhythm การรับจำนวนเต็ม 6 จำนวนทำได้ด้วยคำสั่ง `bd, bm, by, d, m, y = [int(e) for e in input().split()]`

ข้อมูลส่งออก

จำนวนวันตั้งแต่เกิดตามด้วยค่า biorhythm ทาง physical, emotional และ intellectual ตามรูปแบบที่แสดงในตัวอย่าง แสดงจำนวนจริงแบบมีเลข 2 หลักหลังจุดทศนิยม ทำได้ดังนี้ ถ้าต้องการแสดง `x` ก็ใช้คำสั่ง `print( "{:.2f}".format(x) )` และสำหรับการคำนวณค่า biorhythm ตามสูตร ให้ใช้ฟังก์ชัน `sin` และค่า `pi` ใน `math`

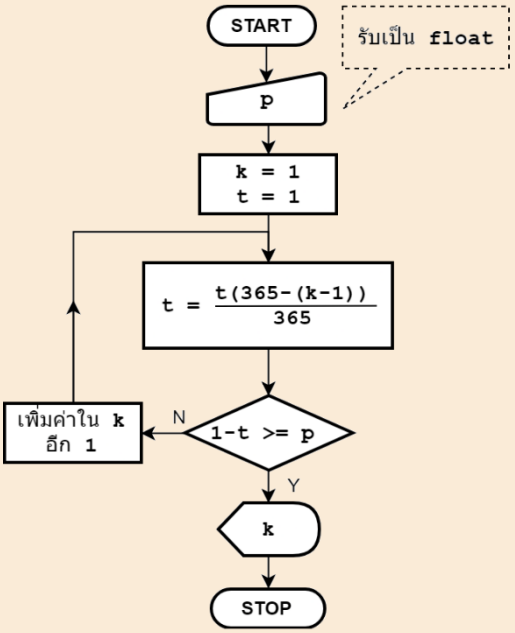
ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
`1 1 2559 1 1 2560`	`366 -0.52 0.43 0.54`
`1 1 2560 1 1 2561`	`365 -0.73 0.22 0.37`
`20 11 2540 10 2 2544`	`1177 0.89 0.22 -0.87`
`10 8 2541 27 10 2559`	`6649 0.52 0.22 0.10`

04: Repetition: for, while

04-01: ผังงานปฏิทรรศน์วันเกิด

จงเขียนโปรแกรมที่ทำงานตามผังงานข้างล่างนี้



ข้อมูลนำเข้า

จำนวนจริงหนึ่งจำนวน

ข้อมูลส่งออก

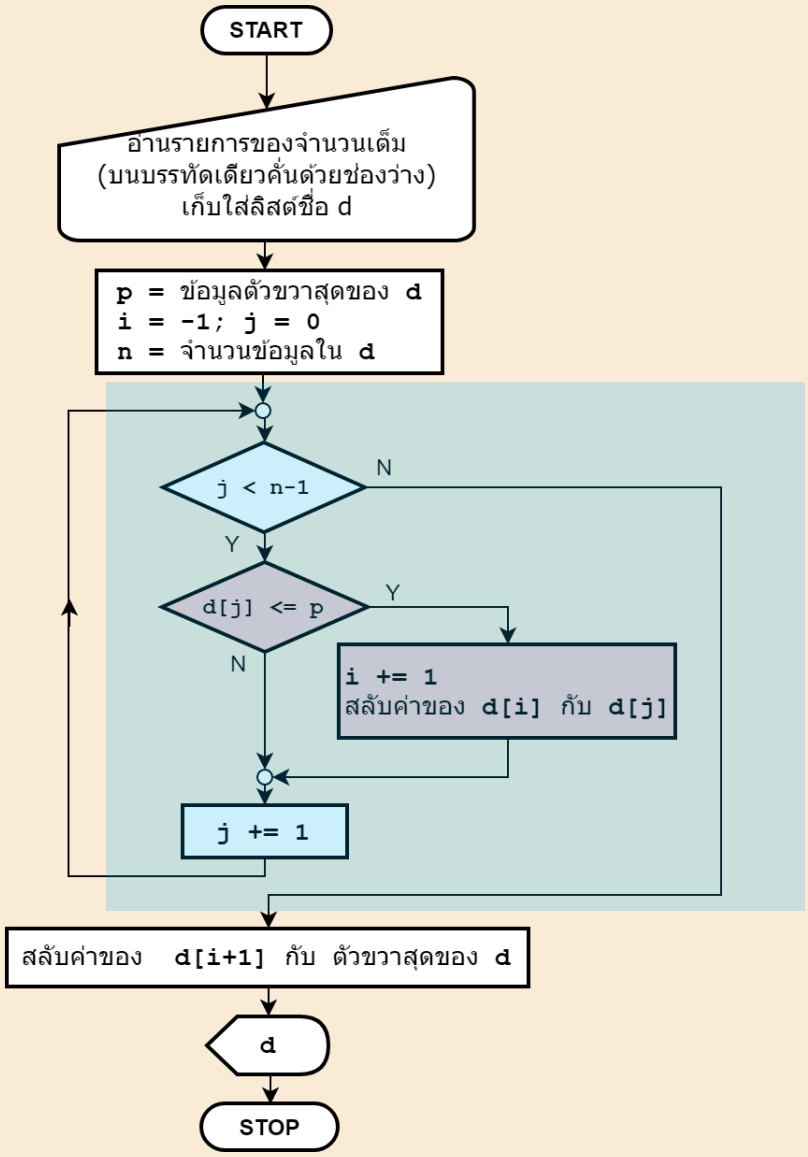
ตามที่แสดงในผังงาน

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
`0.0`	`1`
`0.7`	`30`

04-02: ผังงานการแบ่งส่วน

จงเขียนโปรแกรมที่ทำงานตามผังงานข้างล่างนี้



ข้อมูลนำเข้า

หนึ่งบรรทัดประกอบด้วยรายการของจำนวนเต็มคั่นด้วยช่องว่าง

ใช้คำสั่ง `d = [int(e) for e in input().split()]`

ข้อมูลส่งออก

ตามที่แสดงในผังงาน

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
`1 2 3 4 5`	`[1, 2, 3, 4, 5]`
`5 4 3 2 1`	`[1, 4, 3, 2, 5]`
`9 2 7 1 6 8 4 5`	`[2, 1, 4, 5, 6, 8, 7, 9]`

04-03: ค่าเฉลี่ย

จงเขียนโปรแกรมหาค่าเฉลี่ยของชุดข้อมูลที่รับจากแป้นพิมพ์

ข้อมูลนำเข้า

จำนวนจริงบรรทัดละจำนวน บรรทัดสุดท้ายเป็นตัวอักษร **q**

ข้อมูลส่งออก

ค่าเฉลี่ยของข้อมูลที่รับเข้ามา โดยแสดงเลขหลังจุดทศนิยม 2 ตำแหน่ง

ถ้าไม่มีข้อมูลเลย ให้แสดง **No Data**

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
10 20 30 41.5 q	25.38
10 20 q	15.0
q	No Data

04-04: การประมาณค่าของ $\log_{10} a$ ด้วย bisection (แบบที่ 1)

เราสามารถหาค่าประมาณของ $\sqrt{a}$ ได้ด้วยวิธี bisection ดังนี้

1. ให้ L = 0, U = a
2. เริ่มให้คำตอบอยู่ในช่วง [L, U]
3. x = จุดกึ่งกลางของช่วง
4. ทำข้างล่างนี้ซ้ำ ถ้า x^2 ยังมีค่าไม่ใกล้กับ a ("ใกล้กัน" เมื่อ $|a - x^2| \leq 10^{-10}\max(a, x^2)$)
   - ถ้า $x^2 > a$ ก็เปลี่ยนช่วงเป็น [L, x]
   - ถ้า $x^2 < a$ ก็เปลี่ยนช่วงเป็น [x, U]
   - x = จุดกึ่งกลางของช่วง
5. x คือค่าประมาณของ $\sqrt{a}$

จงนำแนวคิดของ bisection ข้างต้นมาใช้หาค่าประมาณของ $\log_{10} a$ โดยที่ $a \geq 1$

ข้อมูลนำเข้า

จำนวนจริง a (a ที่ใช้ในการทดสอบมีค่าระหว่าง 1 ถึง 600)

ข้อมูลส่งออก

ค่าประมาณของ $\log_{10} a$ โดยแสดงเลขหลังจุดทศนิยม 6 ตำแหน่ง

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
`1`	`0.0`
`100`	`2.0`
`250.0`	`2.39794`
`500.0`	`2.69897`

04-05: การตรวจคำตอบปรนัย

จงเขียนโปรแกรมรับสตริง 2 ตัว สตริงแรกเป็นเฉลยคำตอบปรนัย อีกสตริงเป็นคำตอบของนักเรียน แล้วแสดงว่าถูกกี่ข้อ เช่น

- สตริงเฉลยคือ AAABC แทนเฉลย A, A, A, B และ C ของข้อที่ 1 ถึง 5 ตามลำดับ
- สตริงคำตอบคือ AABCC แทนคำตอบ A, A, B, C และ C ของข้อที่ 1 ถึง 5 ของนักเรียนคนนี้
- สรุปว่า นักเรียนคนนี้ตอบถูก 3 ข้อ

ข้อมูลนำเข้า

บรรทัดแรกเป็นสตริงเฉลย บรรทัดที่สองเป็นสตริงคำตอบ

ข้อมูลส่งออก

จำนวนคำตอบที่ถูกต้อง ถ้าจำนวนข้อของเฉลยไม่ตรงกับของคำตอบ ให้แสดง **`Incomplete answer`**

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
`AAABC` **`AABCC`**	`3`
`AAAAAAAAAAAAAAAAAAAAAA` **`BBBXXBBBBB BBBBB BBBBBB`**	`0`
`AAAAAAAAAAAAAAAAAAAAAAAAAAAAA` **`AAAAAAAAAAA`**	`Incomplete answer`

04-06: วงเล็บเปิดปิด 

จงเขียนโปรแกรมรับสตริง จากนั้นสร้างสตริงใหม่ที่

- แทน (ด้วย [
- แทน [ด้วย (
- แทน) ด้วย]
- แทน] ด้วย)

แล้วแสดงผลทางจอภาพ

ข้อมูลนำเข้า

สตริงหนึ่งบรรทัด

ข้อมูลส่งออก

สตริงที่มีการแทนวงเล็บเปิดปิดตามที่อธิบายไว้ข้างต้น

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
`[[a + (b + [c / d] - e) + f] + 4]`	`((a + [b + (c / d) - e] + f) + 4)`
`no parentheses`	`no parentheses`

04-07: การนับจำนวนคำที่สนใจ

จงเขียนโปรแกรมที่รับคำที่สนใจ แล้วก็รับข้อความหนึ่งบรรทัด จากนั้นนับว่าในข้อความที่รับ มีจำนวนคำที่สนใจกี่คำ

หมายเหตุ: กำหนดให้คำที่สนใจมีแต่ตัวอักษรภาษาอังกฤษ และ ข้อความที่รับเข้ามาประกอบด้วยตัวอักษรภาษาอังกฤษ ตัวเลข หรือเครื่องหมายวรรคตอน `"  (  )  ,  .` หรือ `'`

ข้อมูลนำเข้า

สตริง 2 บรรทัด บรรทัดแรกคือคำที่สนใจ บรรทัดที่สองคือข้อความ

ข้อมูลส่งออก

จำนวนคำที่สนใจในข้อความที่รับเข้ามา (ให้ถือว่าตัวอังกฤษใหญ่ไม่เหมือนตัวเล็ก)

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
`the` `The word "the" is one of the most common words in English.`	`2`
`Sadet` `"Phra Sadet" tham "Phra Sadet" wa ja sadet rue mai sadet.`	`2`

04-08: การวาดสามเหลี่ยมสูง h

จงเขียนโปรแกรมรับจำนวนเต็มที่แทนความสูง h แล้ววาดสามเหลี่ยมหน้าจั่วความสูง h ฐานกว้าง $2h - 1$

ข้อมูลนำเข้า

จำนวนเต็มหนึ่งจำนวนแทนความสูงสามเหลี่ยมหน้าจั่ว (ความสูง ≥ 2)

ข้อมูลส่งออก

สตริงจำนวนบรรทัดเท่ากับความสูงที่ได้รับ แทนรูปสามเหลี่ยมหน้าจั่ว ดังตัวอย่างข้างล่างนี้

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
2	<pre> * ***</pre>
3	<pre> * * * *****</pre>
8	<pre> * * * * * * * * * * * * * * * ***************</pre>

04-09: การประมาณค่าของ $\log_{10}$ a ด้วย bisection (แบบที่ 2)

เราสามารถหาค่าประมาณของ $\log_{10} a$ ได้ด้วย วิธี bisection (อ่านรายละเอียดและตัวอย่างการทำ bisection ในเอกสารประกอบการเรียน) ซึ่งต้องเริ่มกำหนดช่วง [L, U] ที่มั่นใจว่า $\log_{10} a$ อยู่ในช่วงนี้แน่ สำหรับการหา $\log_{10} a$ การให้เริ่มที่ [0, a] จะกว้างเกินไป และอาจเกิดปัญหาในการคำนวณระหว่าง bisection (ลองทำได้ดูได้ และให้ a = 10000.5)

ในที่นี้ขอเสนอวิธีประมาณค่า U ด้วย $1+\lfloor \log_{10} a \rfloor$ ซึ่งมีค่าเท่ากับจำนวนครั้งที่นำ 10 หาร a (แบบปัดเศษทิ้ง) ไปเรื่อย ๆ จนเป็น 0 เช่น a = 120, 120//10 ได้ 12, 12//10 ได้ 1, 1//10 ได้ 0 ซึ่งต้องหาร 3 ครั้ง ก็ให้ U เป็น 3 ก่อนไปทำ bisection

สรุปขั้นตอนการทำงานเป็นดังนี้

1. รับค่า a จากแป้นพิมพ์
2. ให้ L = 0
3. ให้ U มีค่าเท่ากับจำนวนครั้งในการนำ 10 หาร a จนมีค่าเป็น 0 (ข้อแนะนำ: ตรงนี้อาจต้องใช้วงวน while)
4. ใช้ bisection หาค่าประมาณของ $\log_{10} a$ โดยเริ่มที่ช่วง [L, U] จากข้อ 2 กับ 3
5. ให้ทดสอบว่าสองจำนวน a กับ b ใกล้กันเมื่อ $|a-b| \leq 10^{-10}\max(a,b)$)

ข้อมูลนำเข้า

จำนวนจริง a (a นี้มีค่ามากกว่าหรือเท่ากับ 1 แน่ ๆ)

ข้อมูลส่งออก

ค่าประมาณของ $\log_{10} a$ โดยแสดงเลขหลังจุดทศนิยม 6 ตำแหน่ง

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
`1`	`0.0`
`100`	`2.0`
`100000000`	`8.0`
`123456`	`5.091512`

04-10: การแทนชุดข้อมูลด้วย Run-Length Encoding

ถ้ามีสตริงที่มีตัวอักษรซ้ำกันมาก ๆ วางเรียงติด ๆ กัน เช่น "AAAAAAAAAABBBB" ก็อาจแทนด้วย "A 10 B 4" หมายความว่า มี A ติดกัน 10 ตัว ตามด้วย B ติดกัน 4 ตัว เราเรียกการเข้ารหัสทำนองนี้ว่า run-length encoding

จงเขียนโปรแกรมรับสตริงหนึ่งบรรทัด เพื่อเปลี่ยนเป็นสตริงในรูปแบบ run-length encoding แล้วแสดงทางจอภาพ

ข้อมูลนำเข้า

สตริงหนึ่งบรรทัด ประกอบด้วยตัวอักษรภาษาอังกฤษตัวใหญ่เท่านั้น (สตริงที่จะนำมาทดสอบจะเป็นแบบนี้แน่ ๆ)

ข้อมูลส่งออก

สตริงในรูปแบบ run-length encoding ของสตริงที่รับเข้ามา

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
`zzzZZZZZZZZZZZZZZZZZZZZZZZZZZZ`	`z 3 Z 26`
`ABBA`	`A 1 B 2 A 1`

04-11: Zig-Zag / Zag-Zig (แบบที่ 1)

จงเขียนโปรแกรมรับรายการของคู่ข้อมูล X กับ Y และบรรทัดสุดท้ายรับคำสั่งที่อาจเป็น Zig-Zag หรือ Zag-Zig

6 X1 Y1 X2 Y2 X3 Y3 X4 Y4 X5 Y5 X6 Y6 **Zig-Zag** หา min จากข้อมูลเส้นแดง หา max จากข้อมูลเส้นน้ำเงิน	6 X1 Y1 X2 Y2 X3 Y3 X4 Y4 X5 Y5 X6 Y6 **Zag-Zig** หา min จากข้อมูลเส้นแดง หา max จากข้อมูลเส้นน้ำเงิน

- ถ้าเป็น **Zig-Zag**
  - ให้หาค่าน้อยสุดของข้อมูล X1, Y2, X3, Y4, X5, Y6, ... และหาค่ามากสุดของข้อมูล Y1, X2, Y3, X4, Y5, X6, ...
- ถ้าเป็น **Zag-Zig**
  - ให้หาค่าน้อยสุดของข้อมูล Y1, X2, Y3, X4, Y5, X6, ... และหาค่ามากสุดของข้อมูล X1, Y2, X3, Y4, X5, Y6, ...
- แสดงค่าน้อยสุด และค่ามากสุดที่หาได้

ข้อมูลนำเข้า

บรรทัดแรกเป็นจำนวนเต็ม **N** บอกว่าจะมีข้อมูลกี่บรรทัด

N บรรทัดต่อมา แต่ละบรรทัดมีจำนวนเต็มสองจำนวนคั่นด้วยช่องว่าง

บรรทัดสุดท้ายเป็นคำว่า **Zig-Zag** หรือ **Zag-Zig** (ไม่มีคำอื่นแน่ ๆ)

ข้อมูลส่งออก

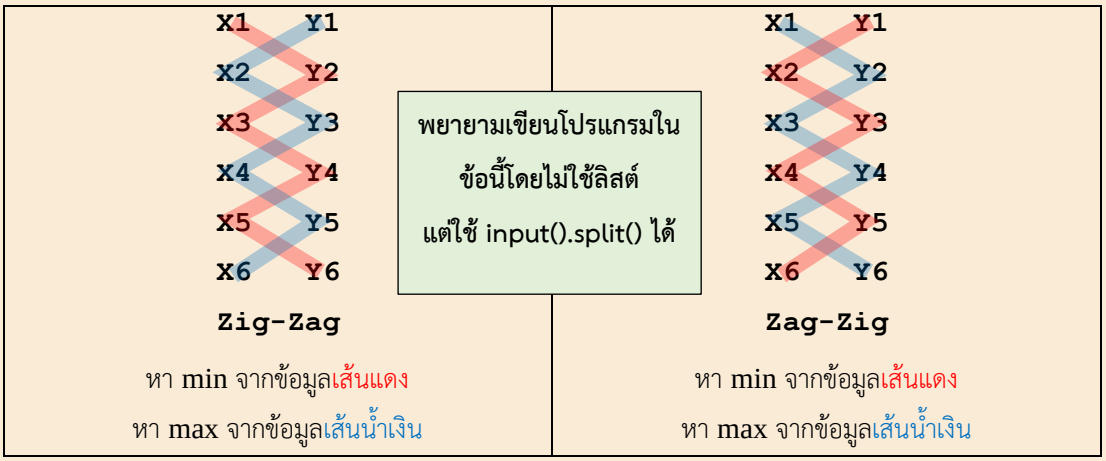
ค่าน้อยสุด และค่ามากสุด (คั่นด้วยช่องว่าง) ตามที่นำเสนอข้างต้น

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
3 -10 10 20 -20 -30 30 Zig-Zag	-30 30
3 -10 10 20 -20 -30 30 Zag-Zig	10 -10

04-12: Zig-Zag / Zag-Zig (แบบที่ 2)

จงเขียนโปรแกรมรับรายการของคู่ข้อมูล X กับ Y และบรรทัดสุดท้ายรับคำสั่งที่อาจเป็น Zig-Zag หรือ Zag-Zig



- ถ้าเป็น **`Zig-Zag`**
  - ให้หาค่าน้อยสุดของข้อมูล X1, Y2, X3, Y4, X5, Y6, ... และหาค่ามากสุดของข้อมูล Y1, X2, Y3, X4, Y5, X6, ...
- ถ้าเป็น **`Zag-Zig`**
  - ให้หาค่าน้อยสุดของข้อมูล Y1, X2, Y3, X4, Y5, X6, ... และหาค่ามากสุดของข้อมูล X1, Y2, X3, Y4, X5, Y6, ...
- แสดงค่าน้อยสุด และค่ามากสุดที่หาได้

ข้อมูลนำเข้า

หลายบรรทัด แต่ละบรรทัดมีจำนวนเต็มสองจำนวนคั่นด้วยช่องว่าง

บรรทัดสุดท้ายเป็นคำว่า **`Zig-Zag`** หรือ **`Zag-Zig`** (ไม่มีคำอื่นแน่ ๆ)

ข้อมูลส่งออก

ค่าน้อยสุด และค่ามากสุด (คั่นด้วยช่องว่าง) ตามที่นำเสนอข้างต้น

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
`-10 10` `20 -20` `-30 30` `Zig-Zag`	`-30 30`
`-10 10` `20 -20` `-30 30` `Zag-Zig`	`10 -10`

05: List Processing

https://www.tvdigitalwatch.com/rating-10-1-65/

05-01: เลขไหนหายไป

จงเขียนโปรแกรมรับสตริง จากนั้นหาและแสดงว่า มีเลข 0 ถึง 9 ใดที่ไม่ปรากฏในสตริงที่รับมา

ข้อมูลนำเข้า

สตริงหนึ่งบรรทัด

ข้อมูลส่งออก

รายการของเลข 0 ถึง 9 ที่ไม่ปรากฏในสตริงที่รับเข้ามา โดยแสดงเลขเรียงจากน้อยไปมาก คั่นด้วยจุลภาค (comma)

ถ้าเลขทุกตัวปรากฏครบในสตริงที่รับ ให้แสดงคำว่า **None**

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
`000099999888765`	`1,2,3,4`
`Arabic numerals are the ten digits:` `0,1,2,3,4,5,6,7,8,9.`	`None`

05-02: ชื่อจริง - ชื่อเล่น

จงเขียนโปรแกรมรับสตริง ถ้าเป็นชื่อจริงก็แสดงชื่อเล่น ถ้าเป็นชื่อเล่นก็แสดงชื่อจริง ถ้าไม่เป็นทั้งชื่อจริงและชื่อเล่น ก็แสดงคำว่า Not found

กำหนดให้ ชื่อจริงและชื่อเล่นที่รู้จักเป็นไปตามที่แสดงในตารางข้างล่างนี้

ชื่อจริง	ชื่อเล่น
Robert	Dick
William	Bill
James	Jim
John	Jack
Margaret	Peggy
Edward	Ed
Sarah	Sally
Andrew	Andy
Anthony	Tony
Deborah	Debbie

พยายามเขียนโปรแกรมนี้โดยไม่ใช้การลุยตรวจ **if...elif...** ไปเรื่อย ๆ เช่นโครงสร้างคำสั่ง

```
if x=="Robert":
  print("Dick")
elif x=="William":
  print("Bill")
...
```

ข้อมูลนำเข้า

บรรทัดแรกเป็นจำนวนเต็ม **N**

อีก **N** บรรทัดตามมาเป็นสตริง

ข้อมูลส่งออก

N บรรทัด แสดงชื่อเล่นหรือชื่อจริงที่คู่กับชื่อจริงหรือชื่อเล่นที่ป้อนเข้ามา บรรทัดละชื่อ ถ้าไม่เป็นชื่อในตารางข้างบนนี้ ก็แสดง **Not found**

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
4 **John** **Jim** **Don** **Debbie**	**Jack** **James** **Not found** **Deborah**

05-03: เพิ่มหลัง - เพิ่มหน้า

จงเขียนโปรแกรมรับจำนวนเต็ม มาเพิ่มในลิสต์ แล้วแสดงลิสต์ที่ได้ทางจอภาพ

ข้อมูลจำนวนเต็มที่รับเข้ามามี 3 ชุด เรียงกันมา แต่ละชุดมีรูปแบบข้อมูลต่างกันดังนี้

1. ชุดแรก เริ่มด้วยบรรทัดแรกที่บอกว่าจะมีข้อมูลอีกกี่ตัวตามมา จากนั้นก็จะมีข้อมูลตามมาเท่านั้นบรรทัด บรรทัดละตัว
2. ชุดสอง เป็นรายการของข้อมูล เรียงจากซ้ายไปขวา คั่นด้วยช่องว่าง
3. ชุดสาม มีข้อมูลบรรทัดละตัว ไปเรื่อย ๆ จนถึงบรรทัดสุดท้ายมีค่า -1 (ค่านี้บอกจุดสิ้นสุดข้อมูล ไม่ใช่ข้อมูลที่ต้องเพิ่มในลิสต์)

การเพิ่มข้อมูลในลิสต์ จะเป็นรูปแบบ เพิ่มต่อท้ายลิสต์ เพิ่มด้านหน้าลิสต์ เพิ่มต่อท้ายลิสต์ เพิ่มด้านหน้าลิสต์ ... สลับแบบนี้ไปเรื่อย ๆ

ข้อมูลนำเข้า

ตามรูปแบบที่นำเสนอข้างต้น

ข้อมูลส่งออก

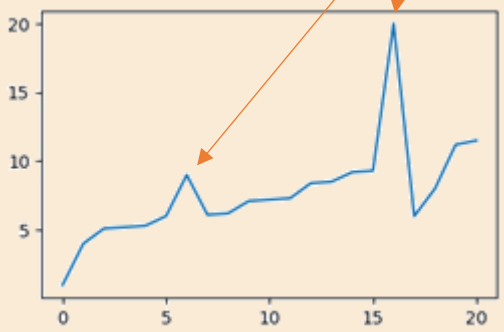
นำลิสต์ที่ได้ แสดงทางจอภาพด้วยคำสั่ง **`print(`** *ชื่อลิสต์* **`)`**

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
4 1 2 3 4 } ชุดที่ 1 11 12 13 14 15 ชุดที่ 2 21 22 23 24 25 -1 } ชุดที่ 3	`[25, 23, 21, 14, 12, 4, 2, 1, 3, 11, 13, 15, 22, 24]`
0 1 2 3 4 5 6 -1	`[6, 4, 2, 1, 3, 5]`
6 1 2 3 4 5 6 -1	`[6, 4, 2, 1, 3, 5]`
0 1 2 3 4 5 6 -1	`[6, 4, 2, 1, 3, 5]`
0 -1	`[]`

05-04: การนับจำนวนยอด

ถ้านำข้อมูลในลิสต์ y = [1, 4, 5.1, 5.2, 5.3, 6, 9, 6.1, 6.2, 7.1, 7.2, 7.3, 8.4, 8.5, 9.2, 9.3, 20, 6, 8, 11.2, 11.5] ไปวาดกราฟเส้นจะได้ดังรูปข้างล่างนี้ จะเห็นว่ามีการเปลี่ยนแปลงของข้อมูลที่มองได้ว่าเป็น "ยอด" คือ ข้อมูลตัวที่เป็นยอด มีค่ามากกว่าทั้งตัวติดกันทางด้านซ้ายและด้านขวา จงเขียนโปรแกรมที่รับรายการของจำนวน แล้วแสดงว่าลิสต์นี้มีอยู่กี่ยอด



ข้อมูลนำเข้า

รายการของจำนวนจริงบนบรรทัดเดียวกันแต่ละตัวคั่นด้วยช่องว่าง

ข้อมูลส่งออก

จำนวนยอดของข้อมูลที่ได้รับ

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
`1 2 3 4 5 6 7 8 9`	`0`
`1 1 1 1 1 1 9 1 1 1 1 1`	`1`
`1 9 1 9 1 9 1 9 1 9 1`	`5`

05-05: จำนวนข้อมูลที่มีค่าต่างกัน

จงเขียนโปรแกรมรับรายการของข้อมูลเพื่อหาว่า มีจำนวนข้อมูลที่มีค่าต่างกันอยู่กี่ค่า คือค่าอะไรบ้าง เช่น 11, 12, 13, 11, 13, 13, 13, 13, 11 มีค่าที่ต่างกันอยู่ 3 ค่า คือ 11, 12, 13

ข้อแนะนำ: หลังจากอ่านข้อมูลเก็บใส่ลิสต์แล้ว ให้ใช้ sort เรียงลำดับข้อมูลในลิสต์จากน้อยไปมาก จะทำให้หาจำนวนข้อมูลที่ต่างกันง่ายขึ้น ด้วยการดูข้อมูลทีละตัวจากซ้ายไปขวาเพียงรอบเดียวเท่านั้น เช่น จากข้อมูลตัวอย่าง 11, 12, 13, 11, 13, 13, 13, 13, 11 หลังจากเรียงลำดับแล้วจะได้ 11, 11, 11, 12, 13, 13, 13, 13, 13 จะเห็นได้ว่า จำนวนค่าที่ต่างกันก็จะเท่ากับจำนวนคู่ที่ติดกันที่มีค่าไม่เท่ากัน บวกอีกหนึ่ง

ข้อมูลนำเข้า

รายการของจำนวนเต็ม

ข้อมูลส่งออก

บรรทัดแรกบอกว่ามีค่าที่ต่างกันกี่ตัว

บรรทัดที่สอง แสดงลิสต์ของค่าที่ต่างกัน (เรียงจากน้อยไปมาก) ขอแค่ 10 ตัวแรกก็พอ (ถ้ามีไม่ถึงก็เอาเท่าที่มี)

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
`1 2 3 9 4 5 6 7 8 0 10 11 14 12 99 98`	`16` `[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]`
`1 9 1 9 1 9 1 9 1 9 1`	`2` `[1, 9]`

05-06: ข้อความคาดการณ์ Collatz

ปัญหา Collatz ถามว่า ถ้าเริ่มที่จำนวนเต็มบวก n การเปลี่ยนแปลงค่า n ด้วยวิธีข้างล่างนี้จะทำให้ในที่สุดแล้ว n เป็น 1 หรือไม่

```
while n ≠ 1:
    if n is even:
        n = n / 2  # ปัดเศษทิ้ง
    else:
        n = 3n + 1
```

แต่เท่าที่ทดสอบกันมา ก็พบว่า n เป็นเต็มบวกอะไร ก็ลงเลยที่ 1 ทั้งนั้น โจทย์ข้อนี้ไม่ได้สนใจเรื่องการพิสูจน์ แต่สนใจให้แสดงการเปลี่ยนแปลงค่า n จนกลายเป็น 1

ข้อมูลนำเข้า

จำนวนเต็มบวก n

ข้อมูลส่งออก

ลำดับการเปลี่ยนแปลงของ n จนเป็น 1 แต่ขอแค่การเปลี่ยนแปลง 15 ค่าสุดท้ายของ n (หรือน้อยกว่าถ้ามีไม่ถึง) โดยแสดงจำนวนต่าง ๆ ที่เปลี่ยนแปลงคั่นด้วย **->** (ดูตัวอย่าง)

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
10	10->5->16->8->4->2->1
18	11->34->17->52->26->13->40->20->10->5->16->8->4->2->1

การเปลี่ยนแปลงกรณี n = **18** คือ
18->9->28->14->7->22->11->34->17->52->26->13->40->20->10->5->16->8->4->2->1
แต่เราขอให้แสดงแค่ 15 ตัวสุดท้าย

05-07: การปรับเกรด

จงเขียนโปรแกรมอ่านรายการของเลขประจำตัวนักเรียนและเกรด ตามด้วยรายการของเลขประจำตัวที่ได้รับการปรับเกรดอีกหนึ่งประจุ (จาก B+ เป็น A, จาก B เป็น B+, จาก C+ เป็น B, ...)

ข้อมูลนำเข้า

แบ่งเป็นสองส่วน

- ส่วนแรกมีหลายบรรทัดเป็นรายการของเลขประจำตัวนักเรียนและเกรด บรรทัดละหนึ่งคน ปิดท้ายบรรทัดสุดท้ายด้วยตัวอักษร **q**
- ส่วนที่สองมีบรรทัดเดียวเป็นรายการของเลขประจำตัว (คั่นด้วยช่องว่าง) ที่ได้รับการปรับเกรดหนึ่งประจุ

ข้อมูลส่งออก

รายการของเลขประจำตัวนักเรียนและเกรด (หลังได้รับการปรับเกรดแล้ว) ของทุกคน คนละบรรทัด ในลำดับเดียวกับที่อ่านในข้อมูลนำเข้า

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
44444 A **22222 F** **11111 B+** **66666 C** **55555 B** **33333 D+** **q** **33333 22222 55555**	**44444 A** **22222 D** **11111 B+** **66666 C** **55555 B+** **33333 C**

ข้อแนะนำ

จากตัวอย่างข้างบนนี้ เราอ่านข้อมูลทั้งหมดเข้ามาสร้างลิสต์ 3 ตัว

- **ids = ['44444', '22222', '11111', '66666', '55555', '33333']**
- **grades = ['A', 'F', 'B+', 'C', 'B', 'D+']**
- **uids = ['33333', '22222', '55555']**
- นำเลขประจำตัวใน **uids** ไปค้นในลิสต์ **ids** ว่าอยู่ที่เลขช่องใด (ด้วยเมท็อด index) ก็ไปเปลี่ยนเกรดที่เลขช่องนั้นในลิสต์ **grades**
  เช่น **'33333'** อยู่ที่ช่อง 5 ของ **ids** เกรดของนักเรียนคนนี้ก็อยู่ที่ช่อง 5 ของ **grades**

05-08: การปรับเกรด (อีกครั้ง)

จงเขียนโปรแกรมอ่านรายการของเลขประจำตัวนักเรียนและเกรด ตามด้วยรายการของเลขประจำตัวที่ได้รับการปรับเกรดอีกหนึ่งประจุ (จาก B+ เป็น A, จาก B เป็น B+, จาก C+ เป็น B, ...

ข้อมูลนำเข้า

แบ่งเป็นสองส่วน

- ส่วนแรกมีหลายบรรทัดเป็นรายการของเลขประจำตัวนักเรียนและเกรด บรรทัดละหนึ่งคน ปิดท้ายบรรทัดสุดท้ายด้วยตัวอักษร `q`
- ส่วนที่สองมีบรรทัดเดียวเป็นรายการของเลขประจำตัว (คั่นด้วยช่องว่าง) ที่ได้รับการปรับเกรดหนึ่งประจุ

ข้อมูลส่งออก

รายการของเลขประจำตัวนักเรียนและเกรด (หลังได้รับการปรับเกรดแล้ว) ของทุกคน คนละบรรทัด ในลำดับเลขประจำตัวเรียงจากน้อยไปมาก

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
44444 A 22222 F 11111 B+ 66666 C 55555 B 33333 D+ q 33333 22222 55555	11111 B+ 22222 D 33333 C 44444 A 55555 B+ 66666 C เรียงตามเลขประจำตัวจากน้อยไปมาก

05-09: จุดที่ใกล้จุดกำเนิดที่สุดเป็นอันดับสาม

จงเขียนโปรแกรมอ่านรายการพิกัดจุดต่าง ๆ เพื่อหาว่าจุดใดใกล้จุดกำเนิดที่สุดเป็นอันดับสาม

ข้อมูลนำเข้า

บรรทัดแรกมีจำนวนเต็มหนึ่งจำนวนที่แทนจำนวนจุด (มีมากกว่าหรือเท่ากับ 3)

หลายบรรทัดตามมา แต่ละบรรทัดมีจำนวจริงสองจำนวน คั่นด้วยช่องว่าง แทนพิกัด x และ y ของจุด หนึ่งบรรทัดหนึ่งจุด

ข้อมูลส่งออก

จุดที่มีระยะถึงจุดกำเนิดใกล้สุดเป็นอันดับสามของจุดทั้งสอง (ข้อมูลทดสอบจะมีระยะถึงจุดกำเนิดไม่เท่ากันเลย) โดยให้แสดงในรูปแบบดังนี้

#*เลขลำดับของจุด***: (***พิกัด x***, ***พิกัด y***)**

หมายเหตุ: จุดแรกที่ในข้อมูลขาเข้ามีเลขลำดับจุดเป็น **1**

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
4 **0.1 0.1** **0.2 0.2** **10.0 10.0** **3.0 3.0**	**#4: (3.0, 3.0)**

ข้อแนะนำ

สร้างลิสต์ที่ภายในเก็บลิสต์ย่อยที่มีองค์ประกอบเป็น **[***ระยะถึงจุดกำเนิด, เลขลำดับของจุด, พิกัด x, พิกัด y***]**

นำไป sort แล้วหยิบช่องที่อินเด็กซ์ 2 ก็ได้คำตอบที่ต้องการ

05-10: ตัดและกรีด

จงเขียนโปรแกรมรับลำดับของไพ่ในสำรับ จากนั้นนำไพ่สำรับนี้มา "ตัด" กับ "กรีด" ตามที่กำหนด

นิยามการตัดไพ่ คือการแบ่งไพ่ในสำรับออกเป็นสองกอง กองละเท่ากัน จากนั้นนำกองหลังมาทับกองหน้า เช่น

ตัดไพ่ A 2 3 4 5 6 7 8 9 10 J Q จะได้ 7 8 9 10 J Q A 2 3 4 5 6

นิยามการกรีดไพ่ คือการแบ่งไพ่ในสำรับออกเป็นสองกอง กองละเท่ากัน จากนั้นนำไพ่ในกองหลังแต่ละใบมาแทรกในไพ่ของกองหน้า เช่น

กรีดไพ่ A 2 3 4 5 6 7 8 9 10 J Q จะได้ A 7 2 8 3 9 4 10 5 J 6 Q

หมายเหตุ: กำหนดให้จำนวนไพ่ในสำรับเป็นจำนวนคู่แน่ ๆ

ข้อมูลนำเข้า

บรรทัดแรกเป็นลำดับของชื่อไพ่จากบนลงล่างในสำรับ (ไม่จำเป็นต้องมี 52 ใบ) เป็นสตริงคั่นด้วยช่องว่าง (ประกอบด้วยสตริงเป็นจำนวนคู่)

บรรทัดที่สองเป็นลำดับการตัดหรือกรีดไพ่ เป็นสตริงที่ประกอบด้วยตัวอักษร ถ้าเป็น **C** คือให้ตัดไพ่ ถ้าเป็น **S** คือให้กรีดไพ่ ถ้าเป็นอักษรอื่นไม่ต้องทำอะไร

ข้อมูลส่งออก

ชื่อไพ่ทั้งหมดจากบนลงล่างในสำรับ หลังจากตัดหรือกรีดตามที่กำหนด

ตัวอย่าง

input (จากแป้นพิมพ์)	คำอธิบาย	output (ทางจอภาพ)
A J Q 10 C	ตัด	Q 10 A J
A J Q 10 CS	ตัด → กรีด	Q A 10 J
A J Q 10 CSC	ตัด → กรีด → ตัด	10 J Q A
A J Q 10 C S C SX	ตัด → กรีด → ตัด → กรีด	10 Q J A

05-11: บัตรคิว

ร้านอาหารแห่งหนึ่งให้บริการสั่งอาหารกลับบ้าน ลูกค้ามาถึงก็กดรับบัตรคิว (**new**) เมื่อพนักงานหน้าร้าน (ซึ่งมีคนเดียว) พร้อมรับออร์เดอร์ ก็จะกดเรียกเบอร์บัตรคิวถัดไป (**next**) ลูกค้าที่มีเบอร์บัตรคิวนั้นก็มาสั่งอาหาร (**order**) เนื่องจากเจ้าของร้านต้องการวิเคราะห์ช่วงเวลาที่ลูกค้าต้องรอตั้งแต่กดบัตรคิวจนถึงเวลาได้สั่งอาหาร จึงเขียนโปรแกรมจัดการบัตรคิวของโจทย์ปัญหานี้

ข้อมูลนำเข้า

- บรรทัดแรกคือค่า `n` ที่เป็นจำนวนเต็มบวกระบุจำนวนบรรทัดคำสั่งที่จะตามมา
- `n` บรรทัดต่อมาเป็นคำสั่งของระบบจัดการบัตรคิว บรรทัดละคำสั่งที่มีรูปแบบดังนี้

คำสั่ง	ความหมาย	ผลลัพธ์ที่แสดงมาทางจอภาพ
`reset n`	ตั้งค่าเริ่มต้นของหมายเลขบัตรคิวใบต่อไปให้เป็น `n` (ทำครั้งเดียวตอนเริ่มต้นเท่านั้น)	ไม่มี
`new t`	ลูกค้ากดบัตรคิว ที่เวลา `t`	**`ticket n`** โดยที่ `n` เป็นหมายเลขบัตรคิวใบล่าสุด (หมายเลขบัตรคิวจะเพิ่มค่าทีละหนึ่งทุกครั้งที่ `new`)
`next`	พนักงานพร้อมรับออร์เดอร์จากลูกค้ารายถัดไป	**`call n`** โดยที่ `n` เป็นหมายเลขบัตรคิวถัดไปที่รอบริการ
`order t`	พนักงานจดออร์เดอร์อาหารจากลูกค้า (ที่เรียกจาก `next` ล่าสุด) ที่เวลา `t`	**`qtime n dt`** โดยที่ `n` คือหมายเลขบัตรที่เรียก `next` ครั้งล่าสุด และ `dt` คือเวลาที่ลูกค้าที่ถือบัตรคิวนี้ต้องรอตั้งแต่ `new` จนถึง `order`
`avg_qtime`	แสดงค่าเฉลี่ยของการรอในแถวคอยของลูกค้าทุกคนที่มารับบริการตั้งแต่โปรแกรมทำงาน (เรียกเมื่อมีการให้บริการแล้วเท่านั้น)	**`avg_qtime x`** โดยที่ **`x`** คือค่าเฉลี่ยของการรอในแถวคอยของลูกค้าทุกคนตั้งแต่โปรแกรมทำงานจนถึงการ order ครั้งล่าสุด (ให้ปัดเศษหลังจุดทศนิยมก่อน แล้วค่อยแสดงผลด้วยคำสั่ง `round(avg,4)` โดย `avg` คือค่าเฉลี่ยที่คำนวณได้)

หมายเหตุ : เวลา **`t`** ทั้งหลายไม่ได้อยู่ในรูปแบบ ชั่วโมง นาที แต่เป็นเลขจำนวนเต็ม (ดูตัวอย่างประกอบ)
คำสั่งที่ได้รับถูกต้องและมีลำดับที่ถูกต้องเป็นไปได้เสมอ เช่น ไม่ต้องกังวลว่า ได้รับ order โดยที่ก่อนนี้ไม่ได้รับ next เป็นต้น

ข้อมูลส่งออก

ผลลัพธ์ที่แสดง ตามตารางที่แสดงข้างบน และดูตัวอย่างประกอบ

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
4 reset 301 new 1100 new 1110 next	ticket 301 ticket 302 call 301
6 reset 301 new 1100 new 1110 next order 1120 avg_qtime	ticket 301 ticket 302 call 301 qtime 301 20 <--- 20 มาจาก 1120 - 1100 avg_qtime 20.0 <--- 20.0 มาจาก 20/1
8 reset 301 new 1100 new 1110 next order 1120 next order 1150 avg_qtime	ticket 301 ticket 302 call 301 qtime 301 20 call 302 qtime 302 40 <--- 40 มาจาก 1150 - 1110 avg_qtime 30.0 <--- 30.0 มาจาก (20+40)/2
14 reset 301 new 1100 new 1110 next order 1120 new 1130 next next order 1160 avg_qtime new 1170 next order 1180 avg_qtime	ticket 301 ticket 302 call 301 qtime 301 20 ticket 303 call 302 <--- พนักงานเรียกเบอร์ 302 แล้วไม่มา call 303 <--- ก็เลยกดเรียกหมายเลขถัดไป 303 qtime 303 30 <--- 30 มาจาก 1160 - 1130 avg_qtime 25.0 <--- 25.0 มาจาก (20+30)/2 ticket 304 call 304 qtime 304 10 avg_qtime 20.0 <--- 20.0 มาจาก (20+30+10)/3

โครงของโปรแกรม

```
ตั้งค่าให้ตัวแปรเสริมที่จำเป็นต้องใช้
q = list()                  # ลิสต์ q ใช้เก็บข้อมูลบัตรคิวที่เหมาะสม
n = int(input())            # อ่านจำนวนคำสั่ง
for k in range(n):
    c = input().split()     # อ่านข้อมูลคำสั่ง
    if c[0] == 'reset':

        ???

    elif c[0] == 'new':

        ???

    elif c[0] == 'next':

        ???

    elif c[0] == 'order':

        ???

    elif c[0] == 'avg_qtime':

        ???
        print( ???, round(???,4) )
```

06: Function

$$(g \circ f)(x) = g(f(x))$$

06-01: การปรับส่วนของโปรแกรมให้เป็นฟังก์ชัน

โปรแกรมข้างล่างนี้รับน้ำหนักและความสูงทางแป้นพิมพ์ เพื่อคำนวณและแสดงค่าพื้นที่ผิวกาย 3 แบบ ตามสูตรทางขวานี้

สูตร Mosteller	$\frac{\sqrt{W \times H}}{60}$
สูตร Du Bois	$0.007184 \times W^{0.425} \times H^{0.725}$
สูตร Fujimoto	$0.008883 \times W^{0.444} \times H^{0.663}$

```
w = float(input()) # body weight
h = float(input()) # body height
mosteller = ((w*h)**0.5) / 60
du_bois = 0.007184 * (w**0.425) * (h**0.725)
fujimoto = 0.008883 * (w**0.444) * (h**0.663)
print("Mosteller =", mosteller)
print("Du Bois =", du_bois)
print("Fujimoto =", fujimoto)
```

จงปรับโปรแกรมข้างบนนี้ใหม่ตามโครงของโปรแกรมข้างล่างนี้ ซึ่งแยกการคำนวณแต่ละสูตรเป็นฟังก์ชัน 3 ฟังก์ชัน และเพิ่มฟังก์ชัน **main** ที่ทำหน้ารับน้ำหนัก ความสูง เรียกใช้ฟังก์ชันทั้งสาม และแสดงผลลัพธ์ เพื่อให้ทำงานเหมือนเดิม

```
def mosteller(w, h):
    # return the body surface area of a person
    # based on body weight (w) and height (h)
    # using Mosteller formula
    ???

def du_bois(w, h):
    # return the body surface area of a person
    # based on body weight (w) and height (h)
    # using Du Bois formula
    ???

def fujimoto(w, h):
    # return the body surface area of a person
    # based on body weight (w) and height (h)
    # using Fujimoto formula
    ???

def main():
    weight = float(input())
    height = float(input())
    ???
    ???
    ???
    print("Mosteller =", round(???, 5))
    print("Du Bois =", round(???, 5))
    print("Fujimoto =", round(???,5))

exec(input()) # DON'T remove this line
```

ข้อมูลนำเข้า

คำสั่งภาษา Python ที่ต้องการให้ทำงาน

ข้อมูลส่งออก

ผลที่ได้จากการสั่งทำงานคำสั่งที่ได้รับ

คำสั่ง `exec( x )` สั่งให้ระบบทำคำสั่งที่เก็บในสตริง `x` เช่น `exec("a = 7")` ก็คือให้ระบบทำคำสั่ง `a = 7`

ดังนั้น `exec(input())` แทนการรับสตริงคำสั่งทางแป้นพิมพ์ แล้วสั่งให้คำสั่งนั้นทำงาน เช่น เมื่อทำงาน แล้วผู้ใช้ป้อน `main()` คำสั่ง `exec(input())` ก็คือ `exec("main()")` คือสั่งให้ฟังก์ชัน `main()` ทำงานนั่นเอง

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
`print(mosteller(56,173))`	`1.6404606399152375`
`print(du_bois(56,173))`	`1.6669772003009131`
`print(fujimoto(56,173))`	`1.6165149017101`
`main()` `56` `173`	`Mosteller = 1.64046` `Du Bois = 1.66698` `Fujimoto = 1.61651`

06-02: การหารากที่สามด้วยเครื่องคิดเลขแบบธรรมดา

จากความรู้พื้นฐานเกี่ยวกับอนุกรมเรขาคณิตที่ว่า $\frac{1}{1-x} = 1 + x + x^2 + x^3 + \ldots$ เมื่อ $|x| < 1$ ถ้าให้ $x = \frac{1}{4}$ จะได้ว่า

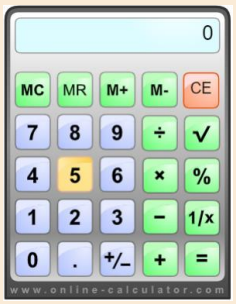
$$\frac{1}{1-1/4} - 1 = \frac{1}{3} = \frac{1}{4^1} + \frac{1}{4^2} + \frac{1}{4^3} + \frac{1}{4^4} + \frac{1}{4^5} + \frac{1}{4^6} + \frac{1}{4^7} + \frac{1}{4^8} + \ldots \quad \text{(สมการที่ 1)}$$

$$= \frac{1}{2^2} + \frac{1}{2^4} + \frac{1}{2^6} + \frac{1}{2^8} + \frac{1}{2^{10}} + \frac{1}{2^{12}} + \frac{1}{2^{14}} + \frac{1}{2^{16}} + \ldots \quad \text{(สมการที่ 2)}$$

$$= \frac{1}{2^2}\left(1 + \frac{1}{2^2}\right)\left(1 + \frac{1}{2^4}\right)\left(1 + \frac{1}{2^8}\right)\ldots \quad \text{(สมการที่ 3)}$$

ใช้สมการที่ 2 หารากที่สามของ y ได้ $y^{\frac{1}{3}} = y^{\frac{1}{2^2} + \frac{1}{2^4} + \frac{1}{2^6} + \frac{1}{2^8} + \frac{1}{2^{10}} + \frac{1}{2^{12}} + \frac{1}{2^{14}} + \frac{1}{2^{16}} + \ldots}$ (สมการที่ 4)

ใช้สมการที่ 3 หารากที่สามของ y ได้ $y^{\frac{1}{3}} = y^{\frac{1}{2^2}(1 + \frac{1}{2^2})(1 + \frac{1}{2^4})(1 + \frac{1}{2^8})\ldots}$ (สมการที่ 5)



จากผลที่ได้ แสดงว่า เราสามารถหารากที่สาม ด้วยเครื่องคิดเลขแบบธรรมดา (ที่มีปุ่ม $\sqrt{x}$ แต่ไม่มีปุ่ม x^y) เช่น $8^{(1/2^2+1/2^4)}$ หาค่าได้ด้วยการกดปุ่มของเครื่องคิดเลขตามลำดับดังนี้ 8 √ √ × 8 √ √ √ √ =

หรือ $8^{(1/2^2)(1+1/2^2)(1+1/2^4)}$ หาค่าได้ด้วยการกดปุ่มตามลำดับดังนี้ 8 √ √ × √ √ × √ √ √ √ =

ตารางข้างล่างนี้ แสดงการใช้สมการที่ 4 กับ 5 เพื่อคำนวณรากที่สามของ 27 ด้วยโปรแกรม Calculator ของ Windows พบว่า การใช้สมการที่ 5 ใช้จำนวนครั้งของการกดปุ่มที่น้อยกว่าการใช้สมการที่ 4 (โดยได้ผลที่มีความแม่นยำพอ ๆ กัน)

ลำดับปุ่มที่กด เพื่อคำนวณตามสมการที่ 4	ลำดับปุ่มที่กด เพื่อคำนวณตามสมการที่ 5
CE MC 2 7 M+ √ 2 ครั้ง × MR √ 4 ครั้ง × MR √ 6 ครั้ง × MR √ 8 ครั้ง × MR √ 10 ครั้ง × MR √ 12 ครั้ง × MR √ 14 ครั้ง × MR √ 16 ครั้ง = ได้คำตอบ 2.999949709941997 (กดปุ่มทั้งสิ้น 92 ครั้ง)	CE 2 7 √ 2 ครั้ง × √ 2 ครั้ง × √ 4 ครั้ง × √ 8 ครั้ง = ได้คำตอบ 2.999949709941997 (กดปุ่มทั้งสิ้น 23 ครั้ง)

ถ้าต้องการคำตอบที่แม่นยำขึ้นก็สามารถกดคำนวณต่อได้ ในตัวอย่างที่แสดงนี้ ขอกดเพื่อให้การคำนวณด้วยสมการ 4 และ 5 ได้ผลลัพธ์ที่มีความแม่นยำพอ ๆ กัน

จงเขียนฟังก์ชันต่าง ๆ ในโปรแกรมข้างล่างนี้

```
def sqrt_n_times(x, n):
    # คืนค่าที่เสมือนการนำค่าใน x มากดปุ่ม √ เป็นจำนวน n ครั้ง

    ???

def cube_root(y):
    # คืนค่าประมาณของรากที่สามของ y โดยใช้วิธีที่เสมือนการกดปุ่มด้วยสูตร
    # y^(1/2^2)(1+1/2^2)(1+1/2^4)(1+1/2^8)(1+1/2^16)(1+1/2^32)
    # ข้อแนะนำ: เรียกใช้ฟังก์ชัน sqrt_n_times

    ???

def main():
    q = float(input())
    print(cube_root(q))

exec(input()) # DON'T remove this line
```

ข้อมูลนำเข้า

คำสั่งภาษา Python ที่ต้องการให้ทำงาน

ข้อมูลส่งออก

ผลที่ได้จากการสั่งทำงานคำสั่งที่ได้รับ

คำสั่ง `exec( x )` สั่งให้ระบบทำคำสั่งที่เก็บในสตริง `x` เช่น `exec("a = 7")` ก็คือให้ระบบทำคำสั่ง `a = 7`

ดังนั้น `exec(input())` แทนการรับสตริงคำสั่งทางแป้นพิมพ์ แล้วสั่งให้คำสั่งนั้นทำงาน เช่น เมื่อทำงาน แล้วผู้ใช้ป้อน `main()` คำสั่ง `exec(input())` ก็คือ `exec("main()")` คือสั่งให้ฟังก์ชัน `main()` ทำงานนั่นเอง

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
`print(sqrt_n_times(10**8,3))`	`10.0`
`print(round(cube_root(27), 4))`	`3.0`
`print(cube_root(5)**3, (5**(1/3))**3)`	`5.000000000000001 4.999999999999998`
`main()` `27`	`2.999999999999999`

06-03: ช่วงเวลา

จงเขียนฟังก์ชันต่าง ๆ ให้ทำงานตามที่เขียนใน comment ของโปรแกรมข้างล่างนี้ (สองฟังก์ชันแรกทำงานถูกต้องแล้ว ไม่ต้องเขียน)

```
def str2hms(hms_str):
    # คืนจำนวนชั่วโมง นาที และวินาที ที่ดึงมาจากสตริง hms
    # เช่น str2hms("10:03:29") ได้ 10,3,29
    t = hms_str.split(':')
    return int(t[0]),int(t[1]),int(t[2])

def hms2str(h,m,s):
    # คืนสตริงในรูปแบบ HH:MM:SS ที่นำจำนวนชั่วโมง นาที และวินาทีมาจาก h,m และ s
    # เช่น hms2str(10,3,29) ได้ "10:03:29"
    return ('0'+str(h))[-2:] + ':' + \
           ('0'+str(m))[-2:] + ':' + \
           ('0'+str(s))[-2:]

def to_sec(h,m,s):
    # คืนจำนวนวินาทีทั้งหมดนับจากเที่ยงคืนจากถึงเวลา h:m:s
    # เช่น to_sec(10,3,29) ได้ 36209
    ???

def to_hms(s):
    # คืนจำนวนชั่วโมง นาที และวินาที ที่หามาจากจำนวนวินาที่ s ทั้งหมดนับจากเที่ยงคืน
    # เช่น to_hms(36209) ได้ 10,3,29
    ???

def diff(h1,m1,s1,h2,m2,s2):
    # คืนจำนวนชั่วโมง นาที และวินาที่ จะเป็นช่วงเวลาตั้งแต่เวลา h1,m1,s1 จนถึง h2,m2,s2
    # เช่น diff(10,57,57, 12,0,0) ได้ 1,2,3
    # หมายเหตุ เวลา h1,m1,s1 ที่ได้รับ ไม่มากกว่า h2,m2,s2 แน่ ๆ
    # (เช่น ไม่มีกรณีให้หาช่วงเวลาตั้งแต่ 23,50,50 ถึง 2,1,1 แน่ ๆ)
    ???

def main():
    # ฟังก์ชันนี้รับเวลาเริ่มต้น และเวลาสิ้นสุด ในรูปแบบ HH:MM:SS
    # เพื่อแสดงช่วงเวลาตั้งแต่เริ่มจนถึงสิ้นสุด ในรูปแบบ HH:MM:SS
    # ดูตัวอย่างในตารางข้างล่าง
    hms_start = input()
    hms_end = input()
    ???

exec(input()) # DON'T remove this line
```

ข้อมูลนำเข้า

คำสั่งภาษา Python ที่ต้องการให้ทำงาน

ข้อมูลส่งออก

ผลที่ได้จากการสั่งทำงานคำสั่งที่ได้รับ

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
`print(to_sec(10,3,29))`	`36209`
`h,m,s = to_hms(36209); print(h,m,s)`	`10 3 29`
`h,m,s = to_hms(36209); print(hms2str(h,m,s))`	`10:03:29`
`dh,dm,ds = diff(10,57,57,12,0,0); print(dh,dm,ds)`	`1 2 3`
`main()` `10:57:57` `12:00:00`	`01:02:03`

คำสั่ง `exec( x )` สั่งให้ระบบทำคำสั่งที่เก็บในสตริง `x` เช่น `exec("a = 7")` ก็คือให้ระบบทำคำสั่ง `a = 7`

ดังนั้น `exec(input())` แทนการรับสตริงคำสั่งทางแป้นพิมพ์ แล้วสั่งให้คำสั่งนั้นทำงาน เช่น เมื่อทำงาน แล้วผู้ใช้ป้อน `main()` คำสั่ง `exec(input())` ก็คือ `exec("main()")` คือสั่งให้ฟังก์ชัน `main()` ทำงานนั่นเอง

06-04: การบวกเลขฐานสอง

Python มีฟังก์ชันให้เรียกใช้มากมาย โจทย์ข้อนี้ให้ฝึกใช้ฟังก์ชัน **`bin(x)`** กับ **`int(x, base)`** ซึ่งมีคำอธิบายหน้าที่ดังนี้

> **`bin(x)`**
> Return a binary string prefixed with **`"0b"`** constructed from an integer **`x`**.

> **`int(x, base)`**
> Return an integer constructed from a string **`x`** in the specified **`base`** (default is 10).

จงเขียนโปรแกรมรับจำนวนเต็มในรูปฐานสอง 2 จำนวน เพื่อหาผลบวกและแสดงในรูปฐานสอง โดยใช้ฟังก์ชัน **`bin`** กับ **`int`** ข้างบน

ข้อมูลนำเข้า

สตริงที่ประกอบด้วยเลข 0 กับ 1 สองชุด คั่นด้วยช่องว่าง แทนจำนวนฐานสอง 2 จำนวน (เป็นจำนวนไม่ติดลบทั้งคู่)

ข้อมูลส่งออก

ผลบวกของจำนวนฐานสองที่รับเข้ามา โดยแสดงผลบวกในรูปฐานสอง

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
`100101 11001`	`111110`
`000000 00000`	`0`
`11111111111111111111111111 1`	`100000000000000000000000000`

06-05: จำนวนเฉพาะถัดไป

ฟังก์ชัน **is_prime(n)** ข้างล่างนี้ตรวจว่า **n** ที่ได้รับเป็นจำนวนเฉพาะหรือไม่ จงใช้ **is_prime** ให้เป็นประโยชน์ในการเขียนฟังก์ชัน **next_prime(n)** และ **next_twin_prim(n)** ตามคำอธิบายที่เขียนใน comment ของแต่ละฟังก์ชัน

```
def is_prime(n):
    # ทดสอบว่า n เป็นจำนวนเฉพาะหรือไม่
    if n <= 1:
        return False
    for k in range(2,int(n**0.5)+1):
        if n%k == 0:
            return False
    return True

def next_prime(N):
    # คืนจำนวนเฉพาะตัวที่มีค่าน้อยสุดที่มากกว่า N



def next_twin_prime(N):
    # คืนจำนวนเฉพาะสองค่าที่เป็น twin prime ที่มีค่าน้อยสุดที่มากกว่า N
    # twin prime คือจำนวนเฉพาะที่มีค่าต่างกัน 2 เช่น 11 กับ 13 หรือ 41 กับ 43



exec(input().strip())      # ต้องมีคำสั่งนี้ ตรงนี้ ตอนส่งให้ Grader ตรวจ
```

ข้อมูลนำเข้า

คำสั่งภาษา Python ที่ใช้ทดสอบการทำงานของฟังก์ชัน

ข้อมูลส่งออก

ผลที่ได้จากการสั่งทำงานคำสั่งที่ได้รับ

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
`print(next_prime(1))`	`2`
`print(next_prime(20))`	`23`
`print(next_prime(10000000))`	`10000019`
`print(next_twin_prime(30))`	`(41, 43)`
`print(next_twin_prime(10000000))`	`(10000139, 10000141)`

06-06: การเรียกใช้ฟังก์ชัน

จงเขียนโปรแกรมอ่านเฉลยและคำตอบของนักเรียน (หลายคน) จากนั้นตรวจให้คะแนนคำตอบและให้เกรดทุกคน แล้วก็นำผลที่ได้ไปเรียงลำดับ ปิดท้ายด้วยการแสดงผลลัพธ์ ข้างล่างนี้คือฟังก์ชันที่เขียนมาให้แล้วจำนวนหนึ่ง โปรแกรมที่จะเขียนสามารถเรียกใช้ฟังก์ชันเหล่านี้ให้ทำงานตามที่ต้องการได้ (หากเรียกใช้ฟังก์ชันให้เหมาะสม ตัวโปรแกรมที่เขียนจะมีคำสั่งไม่น่าเกิน 6 คำสั่งเท่านั้น)

```
def read_answers():
    N = int(input())
    answers = []
    for k in range(N):
        sid, ans = input().split()
        answers.append([sid, ans])
    return answers

def marking(answer, solution):
    score = 0
    for i in range(len(answer)):
        if answer[i] == solution[i]:
            score += 1
    return score

def grading(score):
    g = [[80,"A"], [70,"B"],
         [60,"C"], [50,"D"]]
    for a,b in g:
        if score >= a:
            return b
    return "F"
```

```
def scoring(answers, solution):
    scores = []
    for sid, ans in answers:
        score = marking(ans, solution) /  \
                len(solution) * 100
        grade = grading(score)
        scores.append([sid, score, grade])
    return scores

def report(scores):
    for sid,sc,grade in scores:
        print(sid, sc, grade)

def sort(scores):
    x = []
    for sid,score,grade in scores:
        x.append([score, sid, grade])
    x.sort()
    for i in range(len(x)):
        scores[i] = [x[i][1], x[i][0],
x[i][2]]
```

ข้อมูลนำเข้า

บรรทัดแรกเป็นสตริงที่เก็บลำดับตัวอักษรของเฉลยในแต่ละข้อเรียงกันไป

บรรทัดต่อมาเป็นจำนวนเต็ม N แทนจำนวนนักเรียนที่ส่งคำตอบ

อีก N บรรทัด แต่ละบรรทัดมี 2 สตริงคั่นด้วยช่องว่าง ตัวแรกเป็นเลขประจำตัว อีกตัวป็นสตริงที่เก็บลำดับตัวอักษรของคำตอบ

ข้อมูลส่งออก

รายการของเลขประจำตัว คะแนน และเกรดที่ได้ของนักเรียนแต่ละคน คนละบรรทัด โดยเรียงลำดับตามคะแนนจากมากไปน้อย ในกรณีที่คะแนนเท่ากัน ให้เรียงตามเลขประจำตัวจากมากไปน้อย (คะแนนที่ได้นั้นเป็นร้อยละของคำตอบที่ถูกต้อง)

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
AAAAA 5 0011 ABBBB 2222 AAAAB 3333 AAABB 4444 AAAAA 5555 AAAAB	4444 100.0 A 5555 80.0 A 2222 80.0 A 3333 60.0 C 0011 20.0 F

06-07: สามฟังก์ชันเกี่ยวกับระยะสั้นสุด

จงเขียนฟังก์ชัน **distance1**, **distance2** และ **distance3** ที่ทำงานตามที่เขียนใน comment

```
def distance1(x1, y1, x2, y2):
    # คืนระยะห่างระหว่างจุด (x1,y1) กับ (x2,y2)
    # ตัวอย่างการใช้: d1 = distance1(0.0, 0, 3, 4) ได้ d1 = 5.0


def distance2(p1, p2):
    # p1 และ p2 เป็นลิสต์
    # แต่ละลิสต์แทนจุด ที่เป็นลิสต์ที่ภายในมี 2 ช่อง เก็บพิกัด x กับ y
    # คืนระยะระหว่างจุด p1 กับ p2
    # ตัวอย่างการใช้: d2 = distance2([0.0, 0], [3, 4]) ได้ d2 = 5.0


def distance3(c1, c2):
    # c1 และ c2 แทนวงกลม 2 วง
    # แต่ละลิสต์เป็นลิสต์ 3 ช่อง เก็บพิกัด x กับ y (ของจุดศูนย์กลาง) และรัศมี
    # คืนระยะระหว่างจุดศูนย์กลางของ c1 กับ c2 และค่าจริง/เท็จว่า c1 กับ c2 แตะหรือทับกันหรือไม่
    # ตัวอย่างการใช้: d3,overlap = distance3([0.0, 0, 1], [5, 0, 2])
    #                 ได้ d3 = 5.0, overlap = False


def perimeter(points):
    # points เป็นลิสต์ของจุดต่าง ๆ แต่ละจุดเป็นลิสต์ 2 ช่อง (เก็บพิกัด x และ y)
    #        จุดเหล่านี้คือลำดับของมุมของรูปหลายเหลี่ยม (รูป k เหลี่ยมก็มี k จุด, k>=3)
    # คืนความยาวรอบรูปของรูปหลายเหลี่ยมที่กำหนดโดย points


exec(input().strip())      # ต้องมีคำสั่งนี้ ตรงนี้ ตอนส่งให้ Grader ตรวจ
```

ข้อมูลนำเข้า

คำสั่งภาษา Python ที่ใช้ทดสอบการทำงานของฟังก์ชัน

ข้อมูลส่งออก

ผลที่ได้จากการสั่งทำงานคำสั่งที่ได้รับ

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
`print(distance1(0, 0, 3, 4))`	`5.0`
`print(distance2([0,0], [3,4]))`	`5.0`
`a,b = distance3([0,0,1], [5,0,2]);print(a, b)`	`5.0 False`
`print(perimeter([[0,0], [0,2], [2,2], [2,0]]))`	`8.0`

06-08: อีกสี่ฟังก์ชัน

จงเขียน 4 ฟังก์ชัน ให้ทำงานามที่เขียนอธิบายกำกับแต่ละฟังก์ชัน ในโครงของโปรแกรมข้างล่างนี้

```
def make_int_list(x):
  # รับสตริง x มาแยกและแปลงเป็น int เก็บใน list แล้วคืนเป็นผลลัพธ์
  # เช่น x = '12 34 5' ได้ผลเป็น [12 34 5]


def is_odd(e):
  # คืนค่าจริงเมื่อ e เป็นจำนวนคี่ ถ้าไม่ใช่ คืนค่าเท็จ


def odd_list(alist):
  # คืน list ที่มีค่าเหมือน alist แต่มีเฉพาะตัวที่เป็นจำนวนคี่
  # เช่น alis = [10, 11, 13, 24, 25] จะได้ [11, 13, 25]


def sum_square(alist):
  # คืนผลรวมของกำลังสองของแต่ละค่าใน alist
  # เช่น alist = [1,3,4] ได้ผลเป็น (1*1 + 3*3 + 4*4) = 26


exec(input().strip())  # ต้องมีคำสั่งนี้ ตรงนี้ ตอนส่งให้ Grader ตรวจ
```

ข้อมูลนำเข้า

คำสั่งในการทดสอบฟังก์ชันที่เขียน

ข้อมูลส่งออก

ผลที่ได้จากคำสั่งที่ป้อนเป็นข้อมูลนำเข้า

ตัวอย่าง

input	output (ทางจอภาพ)
`print(make_int_list('1 2 3 4 5'))`	`[1, 2, 3, 4, 5]`
`print(is_odd(3333))`	`True`
`print(odd_list([1,2,3,4,5,6,7]))`	`[1, 3, 5, 7]`
`print(sum_square([1,1,2,3]))`	`15`

06-09: การปรับส่วนของโปรแกรมให้เป็นฟังก์ชัน (อีกข้อ)

โปรแกรมข้างล่างนี้อ่านวันเดือนปีสองวัน แสดงราศีของวันทั้งสอง และแสดงจำนวนวันระหว่างสองวันนี้

```
mname = ["Jan", "Feb", "Mar", "Apr", "May", "Jun",
         "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"]
date1 = input().split()
d1 = int(date1[0])
m1 = mname.index(date1[1][:3]) + 1
y1 = int(date1[2])
# หาราศี
if   d1 >= 22 and m1==3  or d1 <=21 and m1==4  : z1 = "Aries"
elif d1 >= 22 and m1==4  or d1 <=21 and m1==5  : z1 = "Taurus"
elif d1 >= 22 and m1==5  or d1 <=21 and m1==6  : z1 = "Gemini"
elif d1 >= 22 and m1==6  or d1 <=21 and m1==7  : z1 = "Cancer"
elif d1 >= 22 and m1==7  or d1 <=21 and m1==8  : z1 = "Leo"
elif d1 >= 22 and m1==8  or d1 <=21 and m1==9  : z1 = "Virgo"
elif d1 >= 22 and m1==9  or d1 <=21 and m1==10 : z1 = "Libra"
elif d1 >= 22 and m1==10 or d1 <=21 and m1==11 : z1 = "Scorpio"
elif d1 >= 22 and m1==11 or d1 <=21 and m1==12 : z1 = "Sagittarius"
elif d1 >= 22 and m1==12 or d1 <=20 and m1==1  : z1 = "Capricorn"
elif d1 >= 21 and m1==1  or d1 <=20 and m1==2  : z1 = "Aquarius"
elif d1 >= 21 and m1==2  or d1 <=21 and m1==3  : z1 = "Pisces"

date2 = input().split()
d2 = int(date2[0])
m2 = mname.index(date1[1][:3]) + 1
y2 = int(date2[2])

if   d2 >= 22 and m2==3  or d2 <=21 and m2==4  : z2 = "Aries"
elif d2 >= 22 and m2==4  or d2 <=21 and m2==5  : z2 = "Taurus"
elif d2 >= 22 and m2==5  or d2 <=21 and m2==6  : z2 = "Gemini"
elif d2 >= 22 and m2==6  or d2 <=21 and m2==7  : z2 = "Cancer"
elif d2 >= 22 and m2==7  or d2 <=21 and m2==8  : z2 = "Leo"
elif d2 >= 22 and m2==8  or d2 <=21 and m2==9  : z2 = "Virgo"
elif d2 >= 22 and m2==9  or d2 <=21 and m2==10 : z2 = "Libra"
elif d2 >= 22 and m2==10 or d2 <=21 and m2==11 : z2 = "Scorpio"
elif d2 >= 22 and m2==11 or d2 <=21 and m2==12 : z2 = "Sagittarius"
elif d2 >= 22 and m2==12 or d2 <=20 and m2==1  : z2 = "Capricorn"
elif d2 >= 21 and m2==1  or d2 <=20 and m2==2  : z2 = "Aquarius"
elif d2 >= 21 and m2==2  or d2 <=21 and m2==3  : z2 = "Pisces"

days_in_feb1 = 28
if y1 % 400 == 0 or y1 % 100 != 0 and y1%4 == 0 :
    days_in_feb1 = 29

days_in_feb2 = 28
if y2 % 400 == 0 or y2 % 100 != 0 and y2%4 == 0 :
    days_in_feb2 = 29

days_in_m1 = 31
if m1==4 or m1==6 or m1==9 or m1==11 :
    days_in_m1 = 30
elif m1==2 :
    days_in_m1 = days_in_feb1

# เริ่มหาจำนวนวันตั้งแต่ d1,m1,y1 ถึง d2,m2,y2  วิธีที่เขียนอาจผิดพลาดเล็กน้อย ไม่ต้องแก้นะ
days = 0
if m1 < 12 : days += 31
if m1 < 11 : days += 30
if m1 < 10 : days += 31
if m1 < 9  : days += 30
if m1 < 8  : days += 31
if m1 < 7  : days += 31
if m1 < 6  : days += 30
```

```
if m1 < 5  : days += 31
if m1 < 4  : days += 30
if m1 < 3  : days += 31
if m1 < 2  : days += days_in_feb1

if m2 > 1  : days += 31
if m2 > 2  : days += days_in_feb2
if m2 > 3  : days += 31
if m2 > 4  : days += 30
if m2 > 5  : days += 31
if m2 > 6  : days += 30
if m2 > 7  : days += 31
if m2 > 8  : days += 31
if m2 > 9  : days += 30
if m2 > 10 : days += 31
if m2 > 11 : days += 30

days += (days_in_m1 - d1 + 1) + int((y2 - y1 - 1)*365.25) + (d2 - 1)
print(z1, z2)
print(days)
```

โปรแกรมข้างบนนี้มี code ซ้ำกันพอควร จงปรับใหม่โดยใช้ฟังก์ชันตามโครงของโปรแกรมข้างล่างนี้

```
def read_date(): # อ่านวันเดือนปีคั่นด้วยช่องว่าง เดือนเป็นชื่อเดือน คืนลิสต์ เลขวัน เดือน ปี

def zodiac(d,m): # คืนชื่อราศีของวัน d เดือน m

def days_in_feb(y): # คืนจำนวนวันของเดือนกุมภาพันธ์ในปี y

def days_in_month(m,y): # คืนจำนวนวันของเดือน m ในปี y

def days_in_between(d1,m1,y1,d2,m2,y2):
    # คืนจำนวนวันตั้งแต่วันเดือนปี d1,m1,y1 ถึง d2,m2,y2

def main() :
  d1,m1,y1 = read_date()
  d2,m2,y2 = read_date()
  # แสดง ราศีของ d1,m1,y1 กับ ของ d2,m2,y2 บรรทัดเดียกัน คั่นด้วยช่องว่าง
  # แสดงจำนวนวันตั้งแต่ d1,m1,y1 ถึง d2,m2,y2

exec(input().strip())    # ต้องมีคำสั่งนี้ ตรงนี้ ตอนส่งให้ Grader ตรวจ
```

ข้อมูลนำเข้า

คำสั่งในการทดสอบฟังก์ชันที่เขียน

ข้อมูลส่งออก

ผลที่ได้จากคำสั่งที่ป้อนเป็นข้อมูลนำเข้า

ตัวอย่าง

input	output (ทางจอภาพ)
`print(read_date())` `1 Jan 2017`	`[1, 1, 2017]`
`print(zodiac(2,9))`	`Virgo`
`print(days_in_feb(2016))`	`29`
`print(days_in_month(2,2017))`	`28`
`print(days_in_between(1,1,2016, 1,1,2017))`	`366`
`main()` `1 Jan 2015` `2 Aug 2017`	`Capricorn Leo` `943`

07: String and File Processing

https://pixabay.com/illustrations/matrix-matrix-code-control-1274888/

07-01: เอกพจน์พหูพจน์

รูปของคำนามในภาษาอังกฤษมีทั้งแบบเอกพจน์และพหูพจน์ การเขียนคำนามในรูปพหูพจน์จากรูปเอกพจน์มีกฎการเขียนแบบง่ายๆ (ไม่ครอบคลุมทุกกรณี) ดังนี้

- ถ้าเป็นคำนามที่ลงท้าย s, x หรือ ch ทำเป็นพหูพจน์ได้ด้วยการเติม es ต่อท้าย
  (เช่น box → boxes, witch → witches เป็นต้น)
- ถ้าลงท้ายด้วย y แต่ตัวอักษรก่อน y ไม่ใช่สระ, ให้เปลี่ยน y เป็น i แล้วเติม es ต่อท้าย
  (เช่น fly → flies, memory → memories เป็นต้น)
- ถ้าไม่ตรงกับกฎสองข้อข้างบนนี้, ให้ต่อท้ายด้วย s เลย
  (เช่น computer → computers, boy → boys เป็นต้น)

ข้อมูลนำเข้า

คำนามภาษาอังกฤษ 1 คำ ในรูปเอกพจน์

ข้อมูลส่งออก

คำนามภาษาอังกฤษที่รับมาในรูปพหูพจน์

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
box	boxes
fly	flies
boy	boys
computer	computers

07-02: ตัวพิมพ์หลังอูฐ

camelCase เป็นรูปแบบการเขียนวลีที่ไม่มีเครื่องวรรคตอน โดยเปลี่ยนคำแรกเป็นตัวเล็กหมด ส่วนคำที่เหลือเขียนแบบ Capitalization คือตัวแรกของคำเป็นตัวใหญ่ ตัวอื่นเล็กหมด แล้วก็นำทุกคำมาเขียนติดกัน (ตัวเลขคงไว้เหมือนเดิม) เช่น Power of "love" 555 ก็เขียนเป็น powerOfLove555

จงเขียนโปรแกรมที่รับวลีภาษาอังกฤษที่มีตัวอักษรอังกฤษตัวเล็ก ตัวใหญ่ ตัวเลข สัญญลักษณ์พิเศษ และเครื่องหมายวรรคตอนต่าง ๆ จากนั้นขจัดสัญญลักษณ์พิเศษ และเครื่องหมายวรรคตอนต่าง ๆ ออกให้หมด แล้วเปลี่ยนเป็น camelCase เพื่อแสดงทางจอภาพ

ข้อมูลนำเข้า

สตริงหนึ่งบรรทัด

ข้อมูลส่งออก

camelCase ของสตริงที่รับเข้ามา

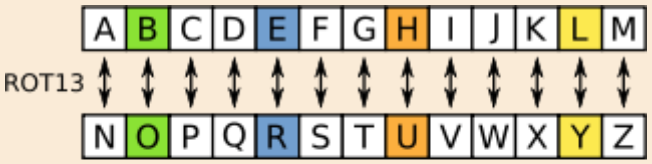
ตัวอย่าง

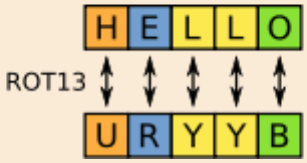
input (จากแป้นพิมพ์)	output (ทางจอภาพ)
`"True" or "False"`	`trueOrFalse`
`My number is 02-000-0000`	`myNumberIs020000000`
`OK OK OK`	`okOkOk`

07-03: การเข้ารหัส ROT-13

From wikipedia (https://en.wikipedia.org/wiki/ROT13):

> ROT13 ("rotate by 13 places", sometimes hyphenated ROT-13) is a simple letter substitution cipher that replaces a letter with the 13th letter after it, in the alphabet. ROT13 is a special case of the Caesar cipher which was developed in ancient Rome.





ข้อมูลนำเข้า

ข้อความหลาย ๆ บรรทัด บรรทัดสุดท้ายเป็นคำว่า end

ข้อมูลส่งออก

ข้อความที่ได้จากการนำข้อความที่รับมาเข้ารหัสแบบ ROT-13

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
`I see trees of green, red roses too` `I see em bloom for me and for you` `And I think to myself,` `"What a wonderful world !"` `end`	`V frr gerrf bs terra, erq ebfrf gbb` `V frr rz oybbz sbe zr naq sbe lbh` `Naq V guvax gb zlfrys,` `"Jung n jbaqreshy jbeyq !"`

07-04: วลีสลับอักษร

Anagram คือ ข้อความที่สร้างได้จากการเรียงสับเปลี่ยนตัวอักษรของอีกข้อความหนึ่ง (ไม่สนใจเครื่องหมายวรรคตอน) เช่น **nat** เป็น anagram ของ **ant** หรือ **William Shakespeare** เป็น anagram ของ **I am a weakish speller** เป็นต้น

จงเขียนโปรแกรมเพื่อตรวจสอบว่า สตริงหนึ่งเป็น anagram ของอีกสตริงหนึ่งหรือไม่

ข้อมูลนำเข้า

สองบรรทัด แต่ละบรรทัดเป็นสตริงที่ประกอบด้วยตัวอักษร ตัวเลข และช่องว่างเว้นวรรค

ข้อมูลส่งออก

ถ้าสตริงทั้งสองที่รับมาเป็น anagram ของกันและกัน ให้แสดง YES แต่ถ้าไม่ใช่ ก็แสดง NO

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
Eleven plus two **Twelve plus one**	**YES**
Elvis **lives**	**YES**
Madam Curie **Radium came**	**YES**
William Shakespeare **I am a weakish speller**	**YES**
Python **Snakes**	**NO**

ข้อแนะนำ

ถ้านำตัวอักษรตัวเลขมาเรียงลำดับ จะตรวจสอบได้ง่ายขึ้น

07-05: น้อยสุด-มากสุด-เฉลี่ย

จงเขียนโปรแกรมหาคะแนนน้อยสุด มากสุด และคะแนนเฉลี่ยของกลุ่มนิสิตในแฟ้มที่เลือกมาเฉพาะนิสิตที่เริ่มเข้าศึกษาในปี พ.ศ. ที่กำหนดให้

ข้อมูลนำเข้า

หนึ่งบรรทัดมีชื่อแฟ้มและปีเริ่มเข้าศึกษาของนิสิตในแฟ้มที่นำมาวิเคราะห์

ข้อมูลส่งออก

คะแนนน้อยสุด คะแนนมากสุด และคะแนนเฉลี่ย ในกรณีที่ไม่มีข้อมูลให้วิเคราะห์ ให้แสดงคำว่า **`No data`**

ตัวอย่าง

ตัวอย่าง: ให้แฟ้มข้อมูลชื่อ **`data.txt`** มีข้อมูลภายในดังแสดงข้างล่างนี้

```
6230012121 90.0
6130351221 80.0
6231027921 80.0
5830548121 65.5
6031087221 79.9
6230550321 70.0
6230432721 60.0
6230215221 50.0
6130518321 70.0
```

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
`data.txt 2562`	`50.0 90.0 70.0`
`data.txt 2561`	`70.0 80.0 75.0`
`data.txt 2560`	`79.9 79.9 79.9`
`data.txt 2559`	`No data`

07-06: ดีเอ็นเอ

สายดีเอ็นเอประกอบด้วยลำดับของนิวคลีโอไทด์ 4 ชนิด แสดงด้วยสตริงของตัวอักษร ‘A’, ‘T’, ‘G’ และ ‘C’ เช่น “AAAACCCGGT” โดยนิวคลีโอไทด์ ‘A’ จะมี ‘T’ เป็นคู่สบ และนิวคลีโอไทด์ ‘G’ จะมี ‘C’ เป็นคู่สบ ถ้าสายดีเอ็นเอมีตัวอักษรอื่นแทรกอยู่ จะถือว่าสายดีเอ็นเอนั้นไม่ถูกต้อง (Invalid DNA) อักษรตัวพิมพ์เล็กและตัวพิมพ์ใหญ่ไม่ต่างกัน ในโจทย์ข้อนี้ ให้นิสิตเขียนโปรแกรมเครื่องประมวลผลสายดีเอ็นเอ

ข้อมูลนำเข้า

บรรทัดแรก สายดีเอ็นเอในรูปแบบของสตริง

บรรทัดที่สอง คือชื่อโอเปอร์เรเตอร์ โดยมีทั้งหมด 3 โอเปอร์เรเตอร์ที่รองรับคือ

1) R คือหา reverse complement ของสายดีเอ็นเอ โดยโอเปอร์เรเตอร์นี้จะทำการเปลี่ยนนิวคลีโอไทด์ในแต่ละลำดับให้เป็นคู่สบของมัน และเมื่อเปลี่ยนเสร็จแล้วต้องกลับด้านสายดีเอ็นเอใหม่นี้ด้วย เช่น AAAACCCGGT จะถูกเปลี่ยนเป็น TTTTGGGCCA และจากนั้นแสดงผลกลับด้านจากขวามาซ้ายเป็น ACCGGGTTTT
2) F คือหาความถี่ของแต่ละนิวคลีโอไทด์ แสดงตามลำดับ A, T, G, C เช่น AAAACCCGGT จะได้ผลลัพธ์เป็น A=4, T=1, G=2, C=3 เป็นต้น (สังเกตว่า มีช่องว่าง 1 ช่องหลังเครื่องหมายคอมมาด้วย !!)
3) D คือการหาจำนวนคู่ของสองนิวคลีโอไทด์ที่อยู่ติดกัน ดังนั้น ถ้าเป็นโอเปอร์เรเตอร์นี้ ต้องมีการรับข้อมูลเพิ่ม อีก 1 บรรทัด คือคู่ของสองนิวคลีโอไทด์ที่สนใจ เช่น GC, AA เป็นต้น โดย GC ไม่เท่ากับ CG และในการนับจำนวนคู่ จะเขียบการหาคู่ไปทีละ 1 นิวคลีโอไทด์ เช่น AAAA แล้วต้องการหา AA จะได้ 3 คู่ เป็นต้น

Hint ให้ตรวจสอบ input เฉพาะกรณีที่เกิด Invalid DNA เท่านั้น รับประกันว่าโอเปอร์เรเตอร์จะถูกต้อง

*** อย่าลืม ต้อง strip() ข้อมูลจาก input() ก่อนนำไปประมวลผล

ข้อมูลส่งออก

ข้อมูลส่งออกมีลักษณะแตกต่างกันไปตามโอเปอร์เรเตอร์ที่ถูกเรียกใช้งาน

หมายเหตุ สีของแต่ละนิวคลีโอไทด์มีไว้เพื่อให้สังเกตแต่ละตัวได้ง่ายขึ้นเท่านั้น

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
ATTTGCGGCATATCC R	GGATATGCCGCAAAT
ATTTGCGGCATATCC F	A=3, T=5, G=3, C=4
ATTTGCGGCATATCC D GC	2
ATTTGCGGCATATCC D TT	2
ATTTGCGGCATTATCC F	Invalid DNA
aTTTgcggCAtaTCC R	GGATATGCCGCAAAT

07-07: รหัสผ่าน

รหัสผ่าน (password) ที่ดีต้องมีลักษณะหลายอย่าง (https://en.wikipedia.org/wiki/Password_strength) ที่ทำให้การเดารหัสผ่านทำได้ยาก ลักษณะสำคัญบางประการของรหัสผ่านที่ดี มีดังนี้ (ในที่นี้ขอพิจารณาเฉพาะตัวอักษรภาษาอังกฤษเท่านั้น)

1. ต้องมีอักขระอย่างน้อย 8 ตัว
2. ต้องมีทั้งตัวอักษรอังกฤษตัวใหญ่ ตัวเล็ก ตัวเลข และสัญลักษณ์อื่น ๆ
3. ต้องไม่มีอักขระ 4 ตัวใดที่ติดกันซ้ำกัน
4. ต้องไม่มีอักขระ 4 ตัวใดที่ติดกันเป็นลำดับตัวเลข ทั้งเพิ่มและลด เช่น **`0123`**, **`2345`**, **`7890`**, **`0987`**, **`...`**
5. ต้องไม่มีอักขระ 4 ตัวใดที่ติดกันเป็นลำดับตัวอักษร ทั้งเพิ่มและลด เช่น **`abcd`**, **`wXYZ`**, **`ZYxW`**, **`...`**
6. ต้องไม่มีอักขระ 4 ตัวใดที่ติดกันเป็นลำดับของปุ่มที่ติดกันบนแป้นพิมพ์ ทางซ้ายไปขวา หรือขวามาซ้าย
   เช่น **`ASDF`**, **`qwer`**, **`REWq`**, **`!@#$`**, **`...`** สำหรับข้อนี้ พิจารณาเฉพาะลำดับปุ่มที่ติดกันแนวนอน ตามแถบสีชมพูในรูปข้างล่างนี้

จงเขียนโปรแกรมเพื่อตรวจรหัสผ่านที่ได้รับว่า ขาดลักษณะสำคัญอะไรบ้าง

ข้อมูลนำเข้า

รหัสผ่านเป็นสตริงหนึ่งบรรทัด

ข้อมูลส่งออก

รายการของข้อความต่าง ๆ ที่ระบุลักษณะสำคัญที่ขาดของรหัสผ่าน (ถ้าขาดหลายลักษณะให้แสดงตามลำดับบนลงล่างข้างล่างนี้อย่างละบรรทัด)

- ถ้าไม่มีลักษณะข้อที่ 1 ให้แสดง **`Less than 8 characters`**
- ถ้าไม่มีลักษณะข้อที่ 2 เรื่องตัวเล็ก ให้แสดง **`No lowercase letters`**
- ถ้าไม่มีลักษณะข้อที่ 2 เรื่องตัวใหญ่ ให้แสดง **`No uppercase letters`**
- ถ้าไม่มีลักษณะข้อที่ 2 เรื่องตัวเลข ให้แสดง **`No numbers`**
- ถ้าไม่มีลักษณะข้อที่ 2 เรื่องสัญลักษณ์ ให้แสดง **`No symbols`**
- ถ้าไม่มีลักษณะข้อที่ 3 ให้แสดง **`Character repetition`**
- ถ้าไม่มีลักษณะข้อที่ 4 ให้แสดง **`Number sequence`**
- ถ้าไม่มีลักษณะข้อที่ 5 ให้แสดง **`Letter sequence`**
- ถ้าไม่มีลักษณะข้อที่ 6 ให้แสดง **`Keyboard pattern`**
- ถ้าครบถ้วนทุกลักษณะ ให้แสดง **`OK`**

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
`pass`	`Less than 8 characters` `No uppercase letters` `No numbers` `No symbols`
`QWERTY1234&`	`No lowercase letters` `Number sequence` `Keyboard pattern`
`abcd9999`	`No uppercase letters` `No symbols` `Character repetition` `Letter sequence`
`ILoveProgrammingInPython2110101$`	`OK`

ข้อแนะนำ

โปรแกรมข้อนี้มีการตรวจสอบหลายแบบ ถ้าแยกเขียนเป็นฟังก์ชัน จะทำให้เขียนอย่างมีระเบียบ และหาที่ผิดได้ง่าย เช่น เขียนตามโครงข้างล่างนี้

```
def no_lowercase(t): # return True if no lowercase, otherwise return False
    ...

def no_uppercase(t):
    ...

def no_number(t):
    ...

def no_symbol(t):
    ...

def character_repetition(t):
    ...

def number_sequence(t):
    ...

def letter_sequence(t):
    ...

def keyboard_pattern(t):
    ...

#-----------------------------
passw = input().strip()
errors = []
if len(passw) < 8:
    errors.append("Less than 8 characters")

if no_lowercase(passw):
    errors.append("No lowercase letters")

if no_uppercase(passw):

...

if len(errors) == 0:
    ...
else:
    ...
```

07-08: การผสานแฟ้มข้อมูล

กำหนดให้ มีแฟ้มข้อมูลสองแฟ้ม แต่ละแฟ้มเก็บข้อมูลนิสิตประกอบด้วย เลขประจำตัวและเกรดเฉลี่ย บรรทัดละคน โดยข้อมูลนิสิตในแฟ้มเรียงลำดับตามคณะ (ดูจากเลขสองตัวท้ายของเลขประจำตัว) และภายในคณะเดียวกันเรียงตามเลขประจำตัวนิสิต เช่น

data1.txt

```
5830548121 2.50
6031087221 3.12
6130351221 3.20
6230432722 2.45
6230550322 3.23
6130518324 3.78
6230215224 2.10
```

data2.txt

```
5930558121 2.30
6231082221 2.12
6030532324 3.87
6030121526 2.99
```

จงเขียนโปรแกรมอ่านข้อมูลจากทั้งสองมาผสานกัน เพื่อแสดงให้เรียงตามคณะ และภายในคณะเรียงตามเลขประจำตัว เช่นจากแฟ้ม data1.txt และ data2.txt ข้างบนนี้ จะได้ผลลัพธ์คือ

```
5830548121 2.50
5930558121 2.30
6031087221 3.12
6130351221 3.20
6231082221 2.12
6230432722 2.45
6230550322 3.23
6030532324 3.87
6130518324 3.78
6230215224 2.10
6030121526 2.99
```

> พยายามเขียนโปรแกรมนี้โดยไม่ต้องใช้ลิสต์ (นอกจากการ split ข้อมูลที่อ่านจากแฟ้ม)

ข้อมูลนำเข้า

ชื่อแฟ้มสองแฟ้มที่เก็บข้อมูลนิสิต ชื่อทั้งสองอยู่ในบรรทัดเดียวกัน คั่นด้วยช่องว่าง

ข้อมูลส่งออก

รายการของเลขประจำตัวและเกรดเฉลี่ยของนิสิตบรรทัดละหนึ่งคน เรียงลำดับตามรูปแบบที่นำเสนอข้างต้น

ตัวอย่าง

input (จากแป้นพิมพ์)		output (ทางจอภาพ)
`data1.txt data2.txt`	แฟ้ม data1.txt `5831111121 2.50` `6032222221 3.12` `6133333321 3.20` `6231111122 2.45` `6232222222 3.23` แฟ้ม data2.txt	`5831111121 2.50` `6032222221 3.12` `6133333321 3.20` `6231111122 2.45` `6232222222 3.23`
`data3.txt data4.txt`	แฟ้ม data3.txt `5841111126 2.77` `6042222226 2.44` `6141111128 3.20` `6232222228 3.99` แฟ้ม data4.txt `5931111121 2.66` `6132222221 2.12` `6231111122 2.13`	`5931111121 2.66` `6132222221 2.12` `6231111122 2.13` `5841111126 2.77` `6042222226 2.44` `6141111128 3.20` `6232222228 3.99`
`data5.txt data6.txt`	แฟ้ม data5.txt `5841111121 2.77` `6042222221 2.44` `6141111128 3.20` `6232222228 3.99` แฟ้ม data6.txt `5931111121 2.66` `6132222221 2.12` `6231111122 2.13`	`5841111121 2.77` `5931111121 2.66` `6042222221 2.44` `6132222221 2.12` `6231111122 2.13` `6141111128 3.20` `6232222228 3.99`

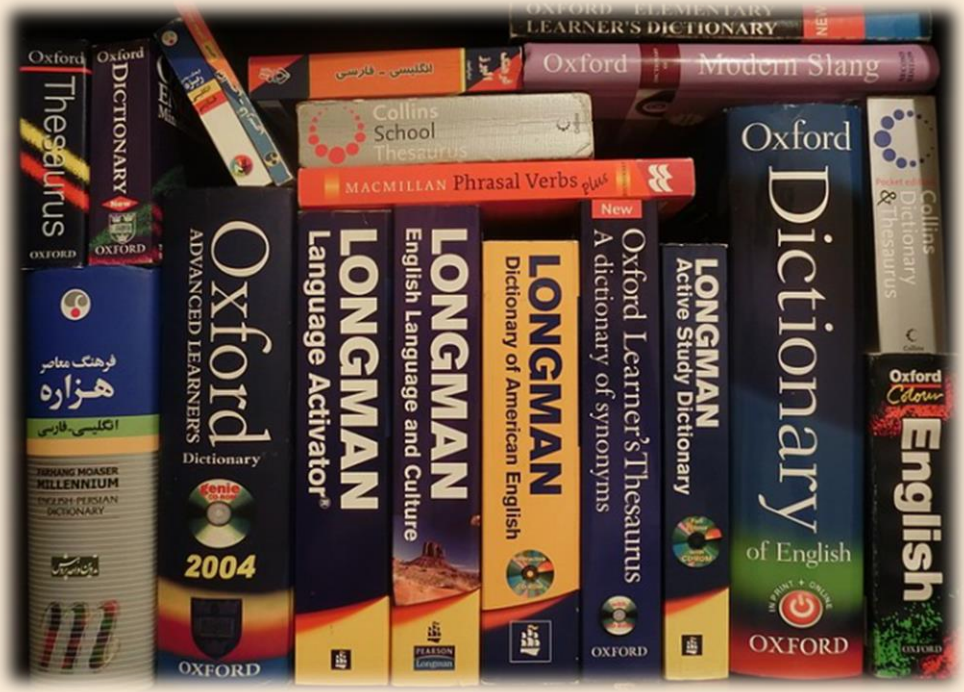
ข้อแนะนำ

การอ่านข้อมูลจากแฟ้มในข้อนี้ อาจต้องใช้คำสั่ง **readline** อ่านทีละบรรทัดเอง การอ่านด้วย **readline** ทุกครั้งจะได้ผลกลับมาเสมอไม่ว่าจะมีข้อมูลจากแฟ้มเหลือให้อ่านหรือไม่ โดยถ้าอ่านแล้วยังมีข้อมูลเหลือให้อ่าน **readline** จะคืนสตริงที่มีความยาวอย่างน้อยหนึ่งอักขระ แต่ถ้าอ่านจนหมดแฟ้ม แล้วไปอ่านด้วย **readline** อีก จะได้สตริงความยาวเป็นศูนย์

สำหรับโจทย์ในข้อนี้ การอ่านข้อมูลจากแฟ้มต้องมีการแยกข้อมูลจากบรรทัดออกเป็น เลขประจำตัว และเกรดเฉลี่ย เกิดขึ้นบ่อย ๆ จึงขอแนะนำให้ใช้ฟังก์ชัน **read_next** ข้างล่างนี้ให้เป็นประโยชน์ (ลองอ่านดูว่า ฟังก์ชันนี้ทำอะไร และใช้งานอย่างไร)

```
def read_next(f):
    while True:
        t = f.readline()
        if len(t) == 0:         # ถ้าอ่านหมดแล้ว ออกจากวงวน
            break
        x = t.strip().split() # ลบ blank ซ้ายขวา
        if len(x) == 2:         # แยกแล้วได้ 2 ส่วน --> ถูกต้อง ก็คืนผล
            return x[0], x[1]
    return "", ""               # แฟ้มหมดแล้ว คืนสตริงว่าง
```

08: Basic Dict



By Alborzagros - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=43326441

08-01: สลับ key กับ value

จงเขียนฟังก์ชัน **revese(d)** ที่รับ **d** เป็น dict ที่ไม่มี value ซ้ำกันเลย มาสร้างและคืน dict ใหม่ที่มี key: value กลับกันกับ **d** และเขียนอีกฟังก์ชัน **keys(d, v)** ที่คืนลิสต์ของคีย์ต่าง ๆ ใน **d** ที่ value มีค่าเป็น **v**

```
def reverse(d):
    # d เป็น dict ที่มี value ไม่ซ้ำกัน
    # คืน dict ใหม่ที่เก็บ key,value ที่ค่าเป็น value,key ของ d ที่ได้รับ



def keys(d, v):
    # คืนลิสต์ที่เก็บค่าของ keys ใน d (เรียงยังไงก็ได้) ที่มีค่า value เท่ากับ v



exec(input().strip())
```

ข้อมูลนำเข้า

คำสั่งภาษา Python ที่ใช้ทดสอบการทำงานของฟังก์ชัน

ข้อมูลส่งออก

ผลที่ได้จากการสั่งทำงานคำสั่งที่ได้รับ

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
`print(reverse({3:"A", 2:"B"}) == {"A":3, "B":2})`	`True`
`print(keys({3:33, 4:33, 5:55, 2:33}, 33))`	`[3, 4, 2]`
`print(keys({3:33, 4:33, 5:55, 2:33}, 9999))`	`[]`

08-02: ชื่อจริง - ชื่อเล่น (อีกแล้ว)

จงเขียนโปรแกรมอ่านรายการชื่อเล่นกับชื่อจริง (กำหนดให้ไม่มีชื่อจริงซ้ำกันเลย และก็ไม่มีชื่อเล่นซ้ำกันด้วย)

จากนั้นรับสตริง ถ้าเป็นชื่อจริงก็แสดงชื่อเล่น ถ้าเป็นชื่อเล่นก็แสดงชื่อจริง ถ้าไม่เป็นทั้งชื่อจริงและชื่อเล่น ก็แสดงคำว่า Not found

ข้อมูลนำเข้า

บรรทัดแรกเป็นจำนวนเต็ม **N**

อีก **N** บรรทัดตามมาเป็นชื่อจริงและชื่อเล่นคั่นด้วยช่องว่าง

บรรทัดที่ **N+2** เป็นจำนวนเต็ม **M**

อีก **M** บรรทัดตามมา ถ้าเป็นชื่อจริงก็คือถามชื่อเล่น ถ้าเป็นชื่อเล่นก็คือถามชื่อจริง

ข้อมูลส่งออก

M บรรทัด แสดงชื่อเล่นหรือชื่อจริงที่คู่กับชื่อจริงหรือชื่อเล่นที่ป้อนเข้ามา บรรทัดละชื่อ ถ้าไม่เป็นชื่อจากข้อมูลนำเข้า ก็แสดง **Not found**

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
10 Robert Dick William Bill James Jim John Jack Margaret Peggy Edward Ed Sarah Sally Andrew Andy Anthony Tony Deborah Debbie 4 John Jim Don Debbie	Jack James Not found Deborah

08-03: การนับตัวอักษร

จงเขียนโปรแกรมรับสตริง เพื่อนับว่ามีตัวอักษรภาษาอังกฤษอะไรบ้าง และมีกี่ตัว (ให้ตัวเล็กเหมือนตัวใหญ่)

ข้อมูลนำเข้า

บรรทัดเดียวเป็นสตริง

ข้อมูลส่งออก

รายการของตัวอักษรที่มีในสตริงที่รับมา ตามด้วยจำนวนตัวที่นับได้ บรรทัดละตัว เรียงตามจำนวนตัวอักษรที่นับได้จากมากมาน้อย ถ้าเท่ากันให้เรียงตามตัวอักษรตามลำดับพจนานุกรม (ดูตัวอย่างประกอบ)

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
AaBbbbbbbCcDddd	**b -> 7** **d -> 4** **a -> 2** **c -> 2**

ข้อแนะนำ

สมมติเรามีลิสต์ `x = [ [ 2,"b"], [ 3,"f"], [ 4,"x"], [ 2,"a"] ]`

แปลงลิสต์ `x` เป็น `[ [-2,"b"], [-3,"f"], [-4,"x"], [-2,"a"]]`

นำไปเรียงจากน้อยไปมากได้ `[ [-4,"x"], [-3,"f"], [-2,"a"], [-2,"b"] ]`

ถ้าเราไม่ดูเครื่องหมายลบ ก็เสมือนเป็นการเรียงข้อมูลใน `x` โดยเรียงจากมากมาน้อยตามตัวแรก และเรียงตามพจนานุกรมตามตัวหลัง

`[ [ 4,"x"], [ 3,"f"], [ 2,"a"], [ 2,"b"] ]`

08-04: ยอดขายไอศกรีม

จงเขียนโปรแกรมอ่านราคาไอศกรีมต่าง ๆ ตามด้วยการขายสินค้าต่าง ๆ ในร้าน

เพื่อหาและแสดงยอดขายรวมของไอศกรีม และไอศกรีมชนิดที่มียอดขายสูงสุด

ข้อมูลนำเข้า

บรรทัดแรกเป็นจำนวนเต็ม N

N บรรทัดต่อมาเป็นข้อมูลราคาขายของไอศกรีม ประกอบด้วย ชื่อไอศกรีม ตามด้วยราคา (จำนวนจริง) คั่นด้วยช่องว่าง

บรรทัดที่ N+2 เป็นจำนวนเต็ม M

M บรรทัดต่อมาเป็นการขายสินค้า ประกอบด้วย ชื่อสินค้า ตามด้วยจำนวนที่ขายได้ คั่นด้วยช่องว่าง

ข้อมูลส่งออก

ถ้าขายไอศกรีมไม่ได้เลย ให้แสดงข้อความว่า **`No ice cream sales`**

ถ้ามีการขายไอศกรีม ให้หาและแสดงยอดขายรวมของไอศกรีมในบรรทัดหนึ่ง และตามด้วยรายการของชื่อไอศกรีมที่มียอดขายสูงสุดอีกบรรทัดหนึ่ง ถ้ามีหลายชื่อให้แสดงชื่อเรียงตามพจนานุกรม (ดูรูปแบบการแสดงในตัวอย่าง)

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
5 Magnum 50 Cornetto 25 PaddlePop 15 AsianDelight 20 Calippo 15 3 Cookie 20 MamaTomYum 3 MangoSheet 10	No ice cream sales
5 Magnum 50 Cornetto 25 PaddlePop 15 AsianDelight 20 Calippo 15 6 Magnum 5 Magnum 5 Cookie 20 MamaTomYum 3 Cornetto 20 AsianDelight 1	Total ice cream sales: 1020.0 Top sales: Cornetto, Magnum

08-05: สมุดหน้าเหลือง

จงเขียนโปรแกรมรับข้อมูลหมายเลขโทรศัพท์ของผู้ใช้บริการซึ่งประกอบด้วย ชื่อ-สกุล และหมายเลขโทรศัพท์
จากนั้นรับคำถามเพื่อค้นหมายเลขโทรศัพท์จากชื่อ-สกุลที่ให้ หรือค้นชื่อ-สกุลจากหมายเลขโทรศัพท์ที่ให้

ข้อมูลนำเข้า

บรรทัดแรกเป็นจำนวนเต็ม N

N บรรทัดต่อมาเป็นข้อมูลผู้ใช้บริการ ประกอบด้วย ชื่อ นามสกุล และหมายเลจโทรศัพท์ คั่นด้วยช่องว่าง

บรรทัดที่ N+2 เป็นจำนวนเต็ม M

M บรรทัดต่อมาเป็นชื่อ นามสกุล (กรณีที่ต้องการให้ค้นหมายเลขโทรศัพท์) หรือ เป็นหมายเลขโทรศัพท์ (กรณีที่ต้องการให้ค้นชื่อ-สกุล)

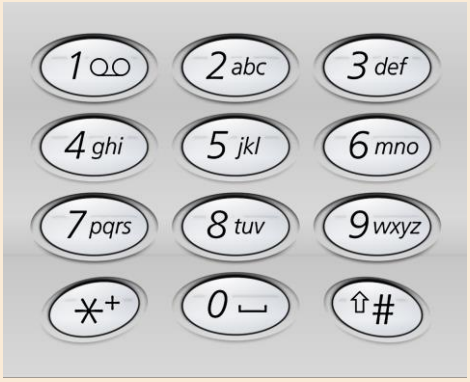
ข้อมูลส่งออก

หมายเลขโทรศัพท์ หรือชื่อ-สกุล ขึ้นกับคำถาม ถ้าหาไม่พบ ให้แสดง **`Not found`**

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
`5` `Anthony Stark 086-111-1111` `Henry Pym 087-222-2222` `Robert Banner 088-333-3333` `Steven Rogers 089-444-4444` `Natasha Romanoff 091-555-5555` `4` `087-222-2222` `Steven Rogers` `Monica Rambeau` `911`	`087-222-2222 --> Henry Pym` `Steven Rogers --> 089-444-4444` `Monica Rambeau --> Not found` `911 --> Not found`

08-06: การป้อนข้อความในโทรศัพท์โบราณ



โทรศัพท์รุ่นที่มีปุ่มกดจะมีตัวอักษรกำกับปุ่มต่าง ๆ ทำให้เราป้อนข้อความได้ ดังแสดงในรูปทางขวานี้ เมื่อต้องการกดตัวอักษร ก็กดปุ่มที่มีตัวอักษรนั้นกำกับ โดยการกดปุ่มหนึ่งครั้ง ก็จะได้ตัวอักษรตัวแรก ถ้ากดเร็ว ๆ สองครั้ง ก็ได้ตัวที่สอง,... เช่น กด **2** จะได้ **a**, กด **22** จะได้ **b**, กด **222** ได้ **c** แต่ถ้าต้องการ **a** สองตัวติดกัน ก็ต้องกด **2** เว้นสักครึ่งวินาที แล้วก็กด **2** อีกครั้ง ถ้าต้องการช่องว่างก็กดปุ่ม **0** ถ้าต้องการสัญลักษณ์พิเศษ หรือเปลี่ยนตัวใหญ่ตัวเล็ก ก็ใช้ปุ่ม ***** กับ **#** (ขอไม่สนใจในที่นี้)

โจทย์ข้อนี้ให้เขียนสองฟังก์ชัน

- **`text2keys( text )`** รับสตริง **`text`** แล้วคืนสตริงตัวเลขและช่องว่าง (ช่องว่างแทนการเว้นช่วงการกด) ที่ต้องกดเพื่อแทนข้อความใน **`text`** เช่น **`text2keys( "Ok, Python" )`** จะได้สตริง **`"666 55 0 7 999 8 44 666 66"`**
  - **`text`** อาจมีตัวอักษรใหญ่หรือเล็กก็ได้ ให้ถือว่าเหมือนกัน
  - ไม่ต้องสนใจอักขระใด ๆ ใน **`text`** ที่ไม่ใช่ตัวอักษรอังกฤษและช่องว่าง
- **`keys2text( keys )`** รับสตริง **`keys`** ที่เก็บเลขและช่องว่าง แล้วคืนสตริงที่เป็นข้อความที่ได้จากการกดเลขตามที่ปรากฏใน **`keys`** เช่น **`keys2text( "666 55 0 7 999 8 44 666 66" )`** จะได้สตริง **`"ok python"`**

```
def text2keys( text ):


def keys2text( keys ):


exec(input().strip())
```

ข้อมูลนำเข้า

คำสั่งภาษา Python ที่ใช้ทดสอบการทำงานของฟังก์ชัน

ข้อมูลส่งออก

ผลที่ได้จากการสั่งทำงานคำสั่งที่ได้รับ

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
`print(text2keys("I am busy."))`	`444 0 2 6 0 22 88 7777 999`
`print(keys2text("444 0 2 6 0 22 88 7777 999"))`	`i am busy`

08-07: เงินสด

กำหนดให้เราแทนเงินด้วย dict ที่มี key เป็นมูลค่าของเหรียญหรือธนบัตร และ value คือจำนวนเหรียญหรือธนบัตรที่มี เช่น **{100: 5, 50: 2, 10: 5, 1: 15}** แทนธนบัตรหนึ่งร้อยบาท 5 ใบ, ห้าสิบ 2 ใบ, สิบบาท 5 ใบ และเหรียญบาท 15 เหรียญ

จงเขียนฟังก์ชันต่าง ๆ ข้างล่างนี้ (ดูตัวอย่างข้างล่างประกอบ)

- **total(pocket)** คืนจำนวนเงินรวมใน **pocket**
- **take(pocket, money)** เติมเงินจาก **money** เข้าใน **pocket** (ทั้งคู่เป็น dict ที่เก็บเงิน)
- **pay(pocket, amt)** ตัดเงินใน **pocket** เป็นจำนวน **amt** (เป็นจำนวนเต็ม) ตัวฟังก์ชันจะคืนเงิน (เป็น dict) ที่จ่ายออกไป ในกรณีที่จ่ายเป็นจำนวน **amt** ไม่ได้ ให้คืน dict ว่าง
  โดยการจ่ายเงิน ใช้วิธีการเลือกธนบัตรหรือเหรียญที่มีในกระเป๋าที่มีมูลค่าสูงสุดที่จ่ายได้ก่อน เช่น {100: 5, 50: 2, 10:5, 1: 15} ต้องการจ่ายออก 57 ก็จะหยิบ 50 ออกหนึ่งใบ และ 1 ออกเจ็ดเหรียญ

```
def total(pocket):

def take(pocket, money_in):

def pay(pocket, amt):

exec(input().strip())
```

ข้อมูลนำเข้า

คำสั่งภาษา Python ที่ใช้ทดสอบการทำงานของฟังก์ชัน

ข้อมูลส่งออก

ผลที่ได้จากการสั่งทำงานคำสั่งที่ได้รับ

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
`p={100:2, 50:2, 5:2, 1:2};print(total(p))`	`312`
`p={100:5};take(p,{100:2, 1:3});print(p)`	`{100: 7, 1: 3}`
`p={100:5};take(p,{100:0, 1:0});print(p)`	`{100: 5, 1: 0}`
`p={10:5, 1:7};print(pay(p, 12));print(p)`	`{10: 1, 1: 2}` `{10: 4, 1: 5}`
`p={10:5, 1:7};print(pay(p, 18));print(p)`	`{}` `{10: 5, 1: 7}` จ่ายไม่ได้
`p={10:5, 1:7};print(pay(p, 100));print(p)`	`{}` `{10: 5, 1: 7}` จ่ายไม่ได้
`p={10:5, 1:7};print(pay(p, 57));print(p)`	`{10: 5, 1: 7}` `{10: 0, 1: 0}` จ่ายหมดกระเป๋า

09: Nested Loop & Nested List

By BrokenSphere - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=3773186

09-01: การเยื้องข้อความออก

จงเขียนโปรแกรมที่สามารถใช้รับ โปรแกรมภาษา Python เพื่อตัดช่องว่างชิดซ้ายออกจำนวนหนึ่ง (เพื่อให้เห็นอย่างชัดเจน สำหรับโจทย์ข้อนี้ขอแทนช่องว่างที่ติดกันเริ่มทางซ้ายด้วยเครื่องหมายจุด) ดังตัวอย่างจากโปรแกรมต้นฉบับทางซ้าย ตัดจุดที่ติดกันทางซ้ายแล้ว จะได้ผลลัพธ์ทางขวา (สังเกตจุดในแถบสีเหลือง ถูกลดลงครึ่งหนึ่ง)

```
n = int(input("n = "))
for k in range(2,n):
....if n%k == 0:
........print("composite..")
........break
else:
....print("prime.")
```

```
n = int(input("n = "))
for k in range(2,n):
..if n%k == 0:
....print("composite..")
....break
else:
..print("prime.")
```

ข้อมูลนำเข้า

บรรทัดแรกเป็นจำนวนเต็ม n ระบุจำนวนบรรทัดที่จะตามมา

บรรทัดที่เป็นสตริงอีก n บรรทัด

กำหนดให้เครื่องหมายจุดที่ติดกันชิดซ้ายของแต่ละบรรทัดที่รับมาจะเป็นจำนวนคู่แน่ ๆ

ข้อมูลส่งออก

สตริงที่รับมา n บรรทัด โดยแต่ละบรรทัดมีเครื่องหมายจุดที่ติดกันชิดซ้ายลดลงจากที่รับมาตามที่กำหนดในโจทย์

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
7 n = int(input("n = ")) for k in range(2,n): ....if n%k == 0: ....print("composite..") ....break else: ....print("prime.")	n = int(input("n = ")) for k in range(2,n): ..if n%k == 0: ....print("composite..") ....break else: ..print("prime.")

09-02: การแยกตัวประกอบแบบง่าย

โจทย์ข้อนี้อยากรู้ว่า จำนวนเต็มบวก N จะถูกแยกเป็นจำนวนประกอบอะไรบ้าง เช่น 200 = $2^3 \times 5^2$ หรือ 3298402 = $2 \times 29^2 \times 37 \times 53$

จงเขียนฟังก์ชัน **`factor(N)`** ที่คืนลิสต์ [[p_1, n_1], [p_2, n_2], ..., [p_k, n_k]] โดยที่ $N = \prod_{i=1}^{k} p_i^{n_i}$ เช่น

- **`factor(200)`** ได้ **`[[2, 3], [5, 2]]`**
- **`factor(3298402)`** ได้ **`[[2, 1], [29, 2], [37, 1], [53, 1]]`**

```
def factor(N):  # N เป็นจำนวนเต็มบวกมากกว่า 1




exec(input().strip())
```

วิธีง่าย ๆ ในการหาตัวประกอบทั้งหมดของ N ทำได้ โดยลุยลองหาร N ด้วย k = 2, 3, ... เมื่อใดพบค่า k ที่หาร N ลงตัว ก็ลุยหารค่า N ด้วย k ไปเรื่อย ๆ จนหารไม่ลงตัว (พร้อมกับนับด้วยว่าหารไปกี่ครั้ง จนหารไม่ลงตัว ค่า k และจำนวนครั้งในการหารนี้ ก็คือส่วนหนึ่งของคำตอบ) แล้วก็ลองค่า k ตัวถัดไป จะพบว่า ค่า k เพิ่ม และค่า N ลด เมื่อใดที่ค่า k เกินค่า N ก็จบการหาตัวประกอบ

เช่น ให้ N = 200 เริ่ม k = 2 พบว่าหารลงตัว หาร N ไปได้ 4 ครั้ง จึงหารไม่ลง ค่า N เปลี่ยน 200 → 100 → 50 → 25 (ได้ [2, 3] เป็นส่วนของคำตอบ) จากนั้นเพิ่ม k = 3 หาร 25 ไม่ลงตัว, k = 4 ก็หารไม่ลงตัว พอ k = 5 ก็หารลงตัว และหารได้ 2 ครั้งค่า N เปลี่ยน 25 → 5 → 1 (ได้ [5, 2] เป็นส่วนของคำตอบ) ถึงตอนนี้ k = 6, N = 1 เป็นอันเสร็จการหาตัวประกอบ ได้คำตอบคือ [[2, 3], [5, 2]] คือ 200 = $2^3 \times 5^2$

ข้อมูลนำเข้า

คำสั่งภาษา Python ที่ใช้ทดสอบการทำงานของฟังก์ชัน

ข้อมูลส่งออก

ผลที่ได้จากการสั่งทำงานคำสั่งที่ได้รับ

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
`print(factor(200))`	`[[2, 3], [5, 2]]`
`print(factor(3298402))`	`[[2, 1], [29, 2], [37, 1], [53, 1]]`
`print(factor(8137740897))`	`[[3, 4], [11, 2], [13, 2], [17, 3]]`

09-03: การคูณเมทริกซ์

เมทริกซ์ของจำนวนจริงขนาด $p \times q$ สามารถแทนได้ด้วยลิสต์ขนาด p ช่องโดยที่แต่ละช่องเก็บลิสต์ขนาด q ช่อง เช่น

$$\begin{bmatrix} 1 & 2 & 3 & 0 \\ 2 & 3 & 0 & 1 \\ 4 & 1 & 2 & 2 \end{bmatrix}$$ แทนได้ด้วย

```
[ [1, 2, 3, 0],
  [2, 3, 0, 1],
  [4, 1, 2, 2] ]
```

จงเขียนฟังก์ชัน

- **`mult_c(c, A)`** ที่คืนเมทริกซ์ที่เป็นผลจากการคูณจำนวนจริง **`c`** กับเมทริกซ์ **`A`**
- **`mult(A, B)`** ที่คืนเมทริกซ์ที่เป็นผลจากการคูณเมทริกซ์ **`A`** กับ **`B`** (ซึ่งคือ **`A`** × **`B`**)

```
def read_matrix():
    m = []
    nrows = int(input())
    for k in range(nrows):
        x = input().split()
        r = []
        for e in x:
            r.append( float(e) )
        m.append(r)
    return m

def mult_c(c, A):



def mult(A, B):



exec(input().strip())
```

From wikipedia (https://en.wikipedia.org/wiki/Matrix_multiplication)

Definition [edit]

If $\mathbf{A}$ is an $n \times m$ matrix and $\mathbf{B}$ is an $m \times p$ matrix,

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nm} \end{pmatrix}, \quad \mathbf{B} = \begin{pmatrix} b_{11} & b_{12} & \cdots & b_{1p} \\ b_{21} & b_{22} & \cdots & b_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ b_{m1} & b_{m2} & \cdots & b_{mp} \end{pmatrix}$$

the *matrix product* $\mathbf{C} = \mathbf{AB}$ (denoted without multiplication signs or dots) is defined to be the $n \times p$ matrix

$$\mathbf{C} = \begin{pmatrix} c_{11} & c_{12} & \cdots & c_{1p} \\ c_{21} & c_{22} & \cdots & c_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ c_{n1} & c_{n2} & \cdots & c_{np} \end{pmatrix}$$

such that

$$c_{ij} = a_{i1}b_{1j} + \cdots + a_{im}b_{mj} = \sum_{k=1}^{m} a_{ik}b_{kj},$$

for $i = 1, \ldots, n$ and $j = 1, \ldots, p$.

ข้อมูลนำเข้า

คำสั่งภาษา Python ที่ใช้ทดสอบการทำงานของฟังก์ชัน

ข้อมูลส่งออก

ผลที่ได้จากการสั่งทำงานคำสั่งที่ได้รับ

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
`A=read_matrix();print(mult_c(0.5,A))` `3` `1 2` `2 3` `3 2`	`[[0.5, 1.0], [1.0, 1.5], [1.5, 1.0]]`
`A=read_matrix();B=read_matrix();print(mult(A,B))` `3` `1 2 3` `1 1 1` `2 2 2` `3` `1 2` `2 3` `3 2`	`[[14.0, 14.0], [6.0, 7.0], [12.0, 14.0]]`

09-04: ปริศนา 15 แผ่น

คงเคยเล่นเกมเลื่อนแผ่นพลาสติกเล็กๆ จำนวน 15 อัน ให้เรียง 1 ถึง 15 จากซ้ายไปขวาบนลงล่าง ดังตัวอย่างในรูปข้างล่างนี้

1	7	2	3
6		8	4
5	9	10	11
13	14	15	12

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	

เราสามารถแทนตารางข้างบนนี้ด้วย **list of lists of ints** (แทนช่องว่างด้วยเลข **0**) เช่น ตารางรูปซ้ายจะแทนด้วย

[[1,7,2,3], [6,0,8,4], [5,9,10,11], [13,14,15,12]]

ให้สังเกตว่า ไม่ใช่ทุกตารางจะสามารถเลื่อนไปสู่เป้าหมายที่ต้องการได้ เช่น แค่สลับ **14** กับ **15** ในตารางทางขวาบนนี้ ซึ่งแทนด้วย **[[1,2,3,4],[5,6,7,8],[9,10,11,12],[13,15,14,0]]** ก็ไม่สามารถเลื่อนไปสู่เป้าหมายได้ (ขอไม่พิสูจน์)

โจทย์ข้อนี้เกี่ยวกับการตรวจตาราง (ในรูปของ **list of lists**) ว่าเป็นตารางที่สามารถเลื่อนไปหาเป้าหมายได้หรือไม่ เราตรวจได้โดยไม่ต้องลงมือเลื่อนหมายเลขในตาราง ดังนี้ (การตรวจสอบนี้ใช้ได้กับตารางที่มีขนาด **nxn** ใดๆ)

1. **flatten:** เปลี่ยน **list of lists of ints** เป็น **list of ints** โดยตัด **0** ทิ้ง เช่น (ขอแสดงกรณีตารางขนาด **3x3**) จาก **[[1,2,0],[3,5,6],[4,7,8]]** ก็เปลี่ยนเป็น **[1,2,3,5,6,4,7,8]**
2. **inversions:** หาจำนวน **inversion** ซึ่งคือจำนวนคู่ของข้อมูลใน **list of ints** ว่า มีกี่คู่ที่ตัวซ้ายมากกว่าตัวขวา เช่น จากตัวอย่างข้างบนนี้ มีข้อมูล **8** ตัว ก็มีทั้งหมด **8*7/2 = 28** คู่ ดังนี้
   (1,2),(1,3),(1,5),(1,6),(1,4),(1,7),(1,8),(2,3),(2,5),(2,6),(2,4),(2,7),(2,8),(3,5),
   (3,6),(3,4),(3,7),(3,8),(5,6),(5,4),(5,7),(5,8),(6,4),(6,7),(6,8),(4,7),(4,8),(7,8)
   ซึ่งมี **2** คู่ที่ตัวซ้ายมากกว่าตัวขวา ดังนั้น จำนวน **inversion** จึงมีค่าเป็น **2**
3. ตารางที่ได้รับ จะเลื่อนได้ไปสู่เป้าหมายได้ ก็เมื่อมีลักษณะ ตรงตามเงื่อนไขข้างล่างนี้

จำนวนแถวของตาราง	จำนวน **inversions**	หมายเลขแถวของตารางที่เลข **0** อยู่ (แถวบนสุดคือแถวที่ **0**)
เลขคี่	เลขคู่	อยู่แถวใดก็ได้
เลขคู่	เลขคี่	เลขคู่
	เลขคู่	เลขคี่

จากตัวอย่างในขั้นตอนที่ **1** ตารางมีจำนวน **3** แถวเป็นเลขคี่ จำนวน **inversions** เป็นเลขคู่ จึงสามารถเลื่อนไปยังเป้าหมายได้ (รายละเอียดอ่านเพิ่มเติมได้ที่ https://www.cs.bham.ac.uk/~mdr/teaching/modules04/java2/TilesSolvability.html)

จงเขียนฟังก์ชันต่าง ๆ ที่ทำงานตาม **comment** ที่เขียนไว้ ในโครงของโปรแกรมข้างล่างนี้

```
def row_number(t, e): # return row number of t containing e   (top row is row #0)

def flatten(t):     # return a list of ints converted from list of lists of ints t

def inversions(x): # return the number of inversions of list x

def solvable(t):    # return True if tiling t (list of lists of ints) is solvable
                    # otherwise return False

exec(input().strip()) # do not remove this line
```

ข้อมูลนำเข้า

คำสั่งในการทดสอบฟังก์ชันที่เขียน

ข้อมูลส่งออก

ผลที่ได้จากคำสั่งที่ป้อนเป็นข้อมูลนำเข้า

ตัวอย่าง

input	output (ทางจอภาพ)
`print(row_number([[0,8,7],[6,5,4],[3,2,1]], 0))`	`0`
`print(flatten([[0,8,7],[6,5,4],[3,2,1]]))`	`[8, 7, 6, 5, 4, 3, 2, 1]`
`print(inversions([8,7,6,5,4,3,2,1]))`	`28`
`print(solvable([[0,8,7],[6,5,4],[3,2,1]]))`	`True`

09-05: จำนวน Primitive Pythagorean Triple

From https://en.wikipedia.org/wiki/Pythagorean_triple

> A **Pythagorean triple** consists of three positive integers *a*, *b*, and *c*, such that $a^2 + b^2 = c^2$. Such a triple is commonly written (*a*, *b*, *c*), and a well-known example is (3, 4, 5). If (*a*, *b*, *c*) is a Pythagorean triple, then so is (*ka*, *kb*, *kc*) for any positive integer *k*. A **primitive Pythagorean triple** is one in which *a*, *b* and *c* are coprime (that is, they have no common divisor larger than 1). A triangle whose sides form a Pythagorean triple is called a Pythagorean triangle, and is necessarily a right triangle

ข้างล่างนี้แสดงโครงของโปรแกรมที่เขียนให้ในโจทย์นี้ โดยเขียนฟังก์ชัน **GCD** ที่หา ห.ร.ม. ให้แล้ว จงเขียนฟังก์ชัน **is_coprime(a, b,c)** และ **primitive_Pythagorean_triples(max_len)** ตามรายละเอียดใน comment

```
def gcd(a,b):
    while b != 0:
        a,b = b, a%b
    return a

def is_coprime(a, b, c):
    # คืนผลการทดสอบว่า a, b และ c เป็น coprime หรือไม่
    # อ่านนิยาม coprime ที่ https://en.wikipedia.org/wiki/Coprime_integers
    ???

def primitive_Pythagorean_triples(max_len):
    # คืนลิสต์ ที่ภายในเก็บลิสต์ย่อยที่มีสมาชิกสามค่าของ a, b และ c
    # โดยที่ a ≤ b ≤ c ≤ max_len
    # ลิสต์ย่อยต่าง ๆ ถูกจัดเรียงตามค่า c จากน้อยไปมาก
    # หากมีค่า c เท่ากัน ให้เรียงตามค่า a เช่น ถ้า max_len = 65 จะได้
    # [[3, 4, 5], [5, 12, 13], [8, 15, 17], [7, 24, 25],
    #  [20, 21, 29], [12, 35, 37], [9, 40, 41], [28, 45, 53],
    #  [11, 60, 61], [16, 63, 65], [33, 56, 65]]

    triple = []
    ???

    return triple

exec(input().strip())
```

ข้อมูลนำเข้า

คำสั่งภาษา Python ที่ใช้ทดสอบการทำงานของฟังก์ชัน

ข้อมูลส่งออก

ผลที่ได้จากการสั่งทำงานคำสั่งที่ได้รับ

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
`print(is_coprime(2,3,6),is_coprime(2,4,8))`	`True False`
`print(primitive_Pythagorean_triples(10))`	`[[3, 4, 5]]`
`print(primitive_Pythagorean_triples(20))`	`[[3, 4, 5], [5, 12, 13], [8, 15, 17]]`

09-06: การบรรจุแบบ First Fit & Best Fit

กำหนดให้มีรายการของจำนวนเต็ม (แต่ละจำนวนมีค่าได้ตั้งแต่ 1 ถึง 100) เช่น [10, 20, 90, 50, 10, 20] คำถามที่น่าสนใจคือ จะแบ่งรายการนี้ออกเป็นรายการย่อย ๆ อย่างไร ที่ทำให้แต่ละรายการย่อยมีผลรวมของจำนวนเต็มไม่เกิน 100 และได้จำนวนรายการย่อยที่น้อยสุด เช่น [10, 20, 90, 50, 10, 10] แบ่งได้ดีสุด คือ [[10, 20, 50, 10], [90, 10]]

ขอนำเสนอวิธีแบ่งแบบง่าย (ที่ไม่ได้จำนวนรายการที่น้อยสุด) โดยพิจารณาข้อมูลทีละตัว แล้วเลือกใส่ในรายการย่อยที่มีอยู่ โดยมีวิธีการเลือกรายการย่อย 2 วิธี

- First Fit วิธีนี้หารายการย่อย (จากซ้ายไปขวา) พบอันที่ใส่ข้อมูลใหม่ได้ ก็ใส่เลย เช่น ต้องการใส่ 20 ลงใน [[90, 5], [50], [70,10]] พบว่า 20 ใส่ใน [90, 5] ไม่ได้ แต่ใส่ใน [50] ได้ ก็ใส่เลย เป็น [[90, 5], [50, 20], [70,10]]
- Best Fit วิธีนี้พิจารณาทุกรายการย่อยที่ใส่ข้อมูลใหม่ได้ แล้วเลือกใส่รายการที่จะทำให้ผลรวมใกล้ 100 ที่สุด เช่น ต้องการใส่ค่า 20 ลงใน [[90, 5], [50], [70,10]] พบว่าใส่ 20 ใน [90, 5] ไม่ได้ แต่ใส่ใน [50] กับ [70,10] ได้ จะเลือกใส่ใน [70,10] เพราะได้ผลที่ใกล้ค่า 100 ที่สุด ได้ผลเป็น [[90, 5], [50], [70,10,20]]

ในกรณีที่ ไม่สามารถหารายการย่อยใดเลยที่ใส่ข้อมูลใหม่ได้ (เพราะใส่แล้วเกินร้อย) ก็สร้างรายการย่อยใหม่ต่อท้ายของที่มีอยู่

จงเขียนสี่ฟังก์ชัน ที่ทำงานตาม comment ที่เขียนข้างล่างนี้

```
def first_fit(L, e):    # นำ e ใส่รายการย่อยใน L ด้วยวิธี first fit


def best_fit(L, e):     # นำ e ใส่รายการย่อยใน L ด้วยวิธี best fit


def partition_FF(D):    # คืนลิสต์ของลิสต์ที่ได้จากการแบ่งข้อมูลใน D ด้วย first fit


def partition_BF(D):    # คืนลิสต์ของลิสต์ที่ได้จากการแบ่งข้อมูลใน D ด้วย best fit


exec(input().strip())
```

ข้อมูลนำเข้า

คำสั่งภาษา Python ที่ใช้ทดสอบการทำงานของฟังก์ชัน

ข้อมูลส่งออก

ผลที่ได้จากการสั่งทำงานคำสั่งที่ได้รับ

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
`L=[[50],[90]];first_fit(L,10);print(L)`	`[[50, 10], [90]]`
`L=[[50],[90]];best_fit(L,10);print(L)`	`[[50], [90, 10]]`
`print(partition_FF([50,90,10,80,50,20]))`	`[[50, 10, 20], [90], [80], [50]]`
`print(partition_BF([50,90,10,80,50,20]))`	`[[50, 50], [90, 10], [80, 20]]`

09-07: การเติมลำดับจำนวนในตาราง

จงเขียนฟังก์ชันข้างล่างนี้ มีหน้าที่สร้างลิสต์ซ้อนลิสต์ที่แทนตาราง ดังตัวอย่างข้างล่างนี้

```
def pattern1(nrows, ncols):
    # nrows ≥ 0, ncols ≥ 0


def pattern2(nrows, ncols):
    # nrows ≥ 0, ncols ≥ 0


def pattern3( N ):  # N ≥ 0




def pattern4( N ):  # N ≥ 0




def pattern5( N ):  # N ≥ 0




def pattern6( N ):  # N ≥ 0



exec(input().strip())
```

pattern1(3,7) ได้

1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21

pattern2(3,7) ได้

1	4	7	10	13	16	19
2	5	8	11	14	17	20
3	6	9	12	15	18	21

pattern3(5) ได้

1	2	3	4	5
0	6	7	8	9
0	0	10	11	12
0	0	0	13	14
0	0	0	0	15

pattern4(5) ได้

1	3	6	10	15
0	2	5	9	14
0	0	4	8	13
0	0	0	7	12
0	0	0	0	11

pattern5(5) ได้

1	6	10	13	15
0	2	7	11	14
0	0	3	8	12
0	0	0	4	9
0	0	0	0	5

pattern6(5) ได้

1	9	10	14	15
0	2	8	11	13
0	0	3	7	12
0	0	0	4	6
0	0	0	0	5

ข้อมูลนำเข้า

คำสั่งภาษา Python ที่ใช้ทดสอบการทำงานของฟังก์ชัน

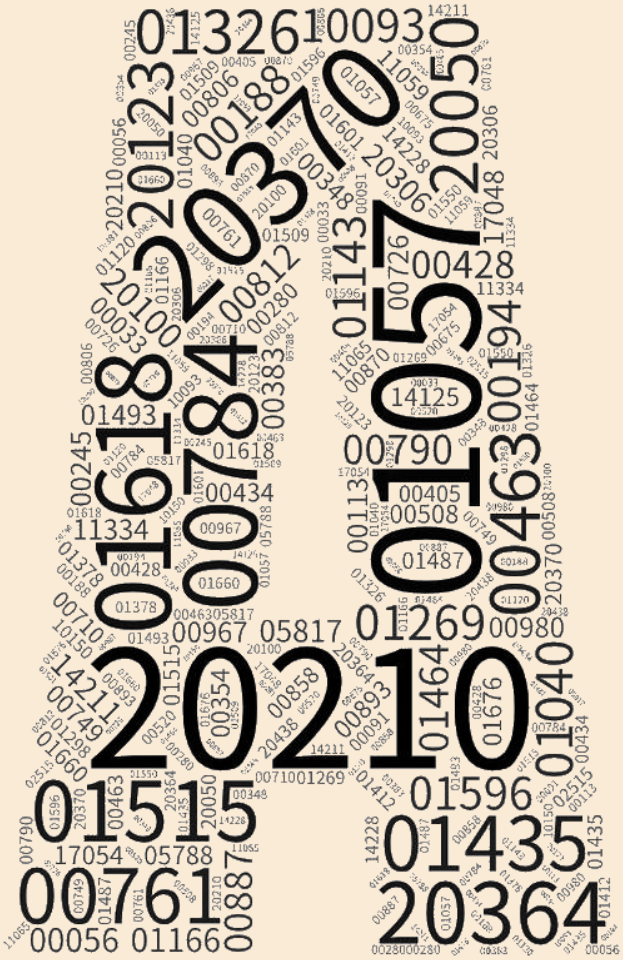
ข้อมูลส่งออก

ผลที่ได้จากการสั่งทำงานคำสั่งที่ได้รับ

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
`print(pattern1(3,4))`	`[[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]]`
`print(pattern2(3,4))`	`[[1, 4, 7, 10], [2, 5, 8, 11], [3, 6, 9, 12]]`
`print(pattern3(4))`	`[[1, 2, 3, 4], [0, 5, 6, 7], [0, 0, 8, 9], [0, 0, 0, 10]]`
`print(pattern4(4))`	`[[1, 3, 6, 10], [0, 2, 5, 9], [0, 0, 4, 8], [0, 0, 0, 7]]`
`print(pattern5(4))`	`[[1, 5, 8, 10], [0, 2, 6, 9], [0, 0, 3, 7], [0, 0, 0, 4]]`
`print(pattern6(4))`	`[[1, 7, 8, 10], [0, 2, 6, 9], [0, 0, 3, 5], [0, 0, 0,`

10: Tuple, Set, Dict



10-01: ยูเนี่ยนและอินเตอร์เซกชัน

จงเขียนโปรแกรมเพื่อหา union และ intersection ของเซตที่กำหนด

ข้อมูลนำเข้า

บรรทัดแรก ระบุจำนวนเต็มบวก n แทนจำนวนเซต

n บรรทัดถัดมา ระบุสมาชิกของเซตของจำนวนเต็ม บรรทัดละหนึ่งเซต (คั่นสมาชิกต่าง ๆ ของเซตด้วยเซตว่าง)

ข้อมูลส่งออก

บรรทัดแรก แสดงขนาดของเซตที่เป็นผลลัพธ์ของการ union ทุกเซต

บรรทัดที่สอง แสดงขนาดของเซตที่เป็นผลลัพธ์ของการ intersection ทุกเซต

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
3 1 2 1 2 3 1 2 1 2 1 2 3 2 5 4 3	5 2
6 100 1000 101 123 200 201 -1 -2 -3	9 0
6 -1 0 1 -1 1 0 0 -1 1 0 1 -1 1 -1 0 1 0 -1	3 3

10-02: ผู้ไม่เคยแพ้ใคร

จงเขียนโปรแกรมเพื่อรับผลการแข่งขันฟุตบอล จากนั้นหาว่า ทีมใดบ้างที่ไม่เคยแพ้เลย

ข้อมูลนำเข้า

บรรทัดแรก ระบุจำนวนเต็มบวก n แทนจำนวนการแข่งขัน

n บรรทัดถัดมา ระบุผลการแข่งขัน โดยระบุทีมที่ชนะตามด้วยทีมที่แพ้ คั่นด้วยเว้นวรรค (ไม่มีกรณีเสมอ)

ข้อมูลส่งออก

มีบรรทัดเดียว แสดงลิสต์ของทีมทั้งหมดที่ไม่แพ้ใคร เรียงตามชื่อทีม โดยอาจใช้คำสั่ง

`print(sorted(winner))` ในการแสดงผล เมื่อ `winner` คือลิสต์ของทีมที่ไม่แพ้ใคร

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
5 **Chelsea Liverpool** **ManU Liverpool** **Liverpool ManU** **Chelsea Arsenal** **Everton ManCity**	**['Chelsea', 'Everton']**
1 **Liverpool WestHam**	**['Liverpool']**
4 **Arsenal ManCity** **Arsenal Everton** **Arsenal Tottenham** **ManCity Arsenal**	**[]**

10-03: ฐานข้อมูล

เราต้องการสร้างระบบสำหรับจัดเก็บข้อมูลผู้สอนรายวิชาต่าง ๆ วิธีที่ง่ายสุดคือ การเก็บรหัสวิชาคู่กับชื่อผู้สอน ดังนี้

```
2110101,Sukree
2110101,Somchai
2100111,Sukree
2110200,Nattee
2110327,Nattee
```

แต่วิธีนี้มีข้อเสียคือ หากผู้สอนเปลี่ยนชื่อ หรือมีการเปลี่ยนรหัสวิชา จะต้องไล่แก้ไขข้อมูลทีละบรรทัด ทำให้การจัดการข้อมูลไม่มีประสิทธิภาพ วิธีการที่ฐานข้อมูลส่วนมากใช้คือ เก็บข้อมูลรหัสวิชาและชื่อผู้สอนแยกออกมา แล้วมีแฟ้มเก็บข้อมูลการจับคู่รหัสวิชากับผู้สอนอีกทีหนึ่ง จากตัวอย่างข้างต้น สามารถเก็บเป็น 3 แฟ้มได้ดังนี้

courses.txt	teachers.txt	database.txt
1,2110101	1,Sukree	1,1
3,2100111	9,Somchai	1,9
4,2110200	5,Nattee	3,1
7,2110327	2,Athasit	4,5
8,2110499		7,5

การเก็บข้อมูลแบบนี้จะทำให้การจัดการข้อมูลมีประสิทธิภาพมากขึ้น เพราะหากต้องแก้ไขข้อมูล สามารถแก้ไขในแฟ้ม courses.txt หรือ teachers.txt แค่บรรทัดเดียว หากต้องการแปลงข้อมูลกลับเป็นรูปแบบที่เราสามารถอ่านแล้วเข้าใจ จะต้องมีการประมวลผลเพิ่มเติม โจทย์ข้อนี้ให้ประมวลผลแฟ้มข้อมูลทั้ง 3 แฟ้ม ให้กลับมาเป็นรูปแบบเดิม

ข้อมูลนำเข้า

มี 3 บรรทัด ระบุชื่อแฟ้มที่เก็บข้อมูลรหัสวิชา ข้อมูลชื่อผู้สอน และแฟ้มจับคู่ระหว่างรหัสวิชากับผู้สอน

ข้อมูลส่งออก

แสดงผลข้อมูลในฐานข้อมูลที่ประมวลผลแล้ว ตามตัวอย่าง ในกรณีที่ข้อมูลผิดพลาด ให้แสดงว่า **record error**

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
courses.txt teachers.txt database.txt	2110101,Sukree 2110101,Somchai 2100111,Sukree 2110200,Nattee 2110327,Nattee
courses.txt teachers.txt database2.txt	record error 2100111,Somchai 2110200,Nattee record error

ให้ **database2.txt** มีข้อมูลดังนี้

```
1,4
3,9
4,5
5,1
```

10-04: เวลารวมตามประเภทเพลง 

มี input เป็นชื่อเพลงตามด้วยชื่อนักร้อง ประเภทเพลง และเวลา จงเขียนโปรแกรมรับประเภทเพลง เพื่อหาเวลารวมของทุกเพลงที่มีประเภทเพลงตามที่กำหนดเรียงจากมากไปน้อย 3 อันดับแรก

ข้อมูลนำเข้า

บรรทัดแรกคือจำนวนเพลง บรรทัดต่อ ๆ มามีจำนวนบรรทัดเท่ากับจำนวนเพลง แต่ละบรรทัดประกอบด้วยชื่อเพลงตามด้วยชื่อนักร้อง ประเภทเพลง และ เวลา (ตามรูปแบบที่แสดงในตัวอย่าง)

ข้อมูลส่งออก

ประเภทเพลง ตามด้วยเวลารวมเป็นนาทีและวินาที เรียงตามลำดับเวลารวม 3 อันดับแรกจากมากมาน้อย บรรทัดละอันดับ (ถ้ามีประเภทเพลงไม่ถึงสามประเภท ก็แสดงเท่าที่มี) หมายเหตุ: ให้ถือว่าเวลารวมของแต่ละประเภทมีไม่เท่ากัน

ตัวอย่าง

input (จากแป้นพิมพ์)

```
9
Shake It Off, Taylor Swift, Pop, 3:39
Rolling In The Deep, Adele, Pop, 3:48
Chandelier, Sia, Pop, 3:36
Roar, Katy Perry, Pop, 3:42
Hotel California, Eagle, Rock, 6:30
We Are the Champions, Queen, Rock, 2:59
Hello Dolly, Louis Armstrong, Jazz, 2:27
Bohemian Rhapsody, Queen, Rock, 5:55
Coward of the County, Kenny Rogers, Country, 4:02
```

output (ทางจอภาพ)

```
Rock --> 15:24
Pop --> 14:45
Country --> 4:02
```

10-05: ตัวการ์ตูน

จงนำข้อมูลชื่อตัวการ์ตูน และชนิดสัตว์ที่ตัวการ์ตูนนั้นเป็น มาสรุปว่า สัตว์ชนิดต่าง ๆ มีชื่อตัวการ์ตูนอะไรบ้าง โดยให้แสดงลำดับของชนิดสัตว์ และลำดับของชื่อตัวการ์ตูนตามลำดับก่อนหลังที่อ่านจากข้อมูลขาเข้า

ข้อมูลนำเข้า

รายการของชื่อตัวการ์ตูนและชนิดของสัตว์คั่นด้วยช่องว่าง บรรทัดละหนึ่งตัว

บรรทัดสุดท้ายเป็นตัวอักษร `q`

ข้อมูลส่งออก

ชนิดของสัตว์ ตามด้วยรายการของชื่อตัวการ์ตูนที่เป็นสัตว์ชนิดนี้ บรรทัดละชนิด

โดยให้แสดงลำดับของชนิดสัตว์ และลำดับของชื่อตัวการ์ตูนตามลำดับก่อนหลังที่อ่านจากข้อมูลขาเข้า

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
Ted, bear Pongo, dog Fozzie, bear Winnie-the-Pooh, bear Nana, dog Hello Kitty, cat Scooby Doo, dog Garfield, cat Yogi, bear Tom, cat Sylvester, cat Pluto, dog Goofy, dog q	bear: Ted, Fozzie, Winnie-the-Pooh, Yogi dog: Pongo, Nana, Scooby Doo, Pluto, Goofy cat: Hello Kitty, Garfield, Tom, Sylvester

10-06: ใครเคยไปที่ที่อีกคนใครไป

จากการวิเคราะห์ข้อมูลใน **Social network** ซึ่งเก็บข้อมูลการเดินทางไปยังเมืองต่าง ๆ ของผู้ใช้ในรูปแบบ

ID: x1, x2, x3, … เมื่อ **x1, x2, x3, …** เป็นชื่อเมือง

ให้เขียนโปรแกรมเพื่ออ่านข้อมูลจากแป้นพิมพ์ และรับ **ID** เข้าเป็น **keyID** 1 อัน จากนั้นให้คำนวณและแสดงผลลัพธ์ **ID** ทั้งหมดที่เคยไปเมืองเดียวกับ **keyID** ที่รับเข้ามา ให้แสดงผลลัพธ์เป็นลิสต์ของ **ID** เรียงเป็นบรรทัด บรรทัดละหนึ่ง **ID** ตามลำดับ **ID** ที่รับเข้ามา ถ้าไม่มีใครที่เคยไปเมืองเดียวกับ **keyID** เลย ให้แสดง **Not Found**

ข้อมูลนำเข้า

บรรทัดแรกเป็นจำนวนผู้ใช้ n

n บรรทัดต่อมาเป็นข้อมูลการเดินทางไปยังเมืองต่าง ๆ ของผู้ใช้ กำหนดให้ ผู้ใช้ทุกคนเคยไปอย่างน้อย 1 เมือง

บรรทัดสุดท้ายระบุ **keyID** ที่ต้องการค้นหา

ข้อมูลส่งออก

รายการของ **ID** บรรทัดละหนึ่ง **ID** เรียงตามลำดับ **ID** ที่รับเข้ามา

ตัวอย่าง

input	output (ทางจอภาพ)
6 51234621: A, B, D, E, F 427613829: B, D, G, H, I 38216542: Z, B, D, J 423212822: AA, B1, C3, D 4126548: J, Z3 98871973331: Q, M, N 4126548	38216542
6 51234621: A, B, D, E, F 427613829: B, D, G, H, I 38216542: Z, B, D, J 423212822: AA, B1, C3, D 4126548: J, Z3 98871973331: Q, M, N 423212822	51234621 427613829 38216542
6 51234621: A, B, D, E, F 427613829: B, D, G, H, I 38216542: Z, B, D, J 423212822: AA, B1, C3, D 4126548: J, Z3 98871973331: Q, M, N 98871973331	Not Found

10-07: ตามหาดาวเด่น

เรานิยามให้ “ดาวเด่น” ในงานเลี้ยง คือบุคคลที่ทุกคนในงานรู้จักเขา แต่เขาไม่รู้จักคนอื่นในงานเลย **(รู้จักตัวเองก็ไม่ว่ากัน)**

กำหนดให้ข้อมูลที่ได้รับ จะบอกว่า ใครรู้จักใครบ้าง เช่น **Ploy Pat** หมายความว่า **Ploy** รู้จัก **Pat** (แต่ไม่ได้แปลว่า **Pat** รู้จัก **Ploy**) ดังนั้น ถ้าข้อมูลขาทั้งหมดคือ

```
Ploy Pat
Ploy Boy
Eak Pat
Boy Pat
Poom Pat
Boy Eak
```

สรุปได้ว่า ทุกคน (ยกเว้น **Pat**) รู้จัก **Pat** และ **Pat** ไม่รู้จักใคร แสดงว่า **Pat** เป็นเซเลบ

โปรแกรมข้างล่างนี้ใช้ **dict** ชื่อว่า **R** มี **key** เป็นชื่อ และ **value** เก็บ **set** ของชื่อที่ **key** รู้จัก ดังนั้น จากข้อมูลข้างบนนี้ย่อมได้

```
R = { 'Ploy': {'Boy', 'Pat'},
      'Boy' : {'Pat', 'Eak'},
      'Pat' : set(),
      'Poom': {'Pat'},
      'Eak' : {'Pat'}
    }
```

จงเขียนฟังก์ชัน **knows**, **is_celeb** และ **find_celeb** ข้างล่างนี้ให้สมบูรณ์ตาม **comment** และเก็บข้อมูลที่อธิบายข้างบนนี้

```
def knows(R,x,y):
    # return True if x knows y

def is_celeb(R,x): # return True if a is celeb, otherwise return False
    # return False if x knows someone who is not him/herself
    # return False if there exists someone in R who don't know x
    # otherwise return True

def find_celeb(R):
    # for each person x in the party
    #   if x is celeb --> return x
    # if no celeb in the party --> return None

def read_relations() :
    # build a dictionary R from inputs
    # whose structure is shown in the example

    R = dict()
    while True:
        d = input().split()
        if len(d) == 1 : break

        ???

    return R

def main():
    R = read_relations()
    c = find_celeb(R)
    if c == None :
      print('Not Found')
    else:
      print(c)

exec(input().strip())  # do not remove this line
```

ข้อมูลนำเข้า

คำสั่งในการทดสอบฟังก์ชันที่เขียน

ข้อมูลส่งออก

ผลที่ได้จากคำสั่งที่ป้อนเป็นข้อมูลนำเข้า

ตัวอย่าง

input	output (ทางจอภาพ)
main() Ploy Pat Ploy Boy Eak Pat Boy Pat Poom Pat Boy Eak q	Pat
main() Ploy Pat Ploy Boy Eak Pat Boy Pat Poom Pat Boy Eak Noo-sa Tim q	Not Found

10-08: การบวกและการคูณพหุนาม

จาก Wikipedia (https://en.wikipedia.org/wiki/Polynomial)

Polynomial

From Wikipedia, the free encyclopedia

In mathematics, a **polynomial** is an expression consisting of variables (also called indeterminates) and coefficients, that involves only the operations of addition, subtraction, multiplication, and non-negative integer exponents of variables. An example of a polynomial of a single indeterminate, x, is $x^2 - 4x + 7$. An example in three variables is $x^3 + 2xyz^2 - yz + 1$.

โจทย์ข้อนี้เกี่ยวกับการหาผลบวกของ polynomial แบบตัวแปรเดียว 2 ชุด เราสามารถแทน polynomial ด้วยลิสต์ของทูเปิล แต่ละทูเปิลมีสมาชิก 2 ตัว ตัวแรกคือสัมประสิทธิ์ ตัวหลังคือเลขชี้กำลัง เช่น $4x^2 + 3x - 1$ แทนด้วย [(4, 2), (3, 1), (-1, 0)] โดยเก็บทูเปิลในลิสต์เรียงลำดับตามเลขชี้กำลังจากมากไปน้อย จงเขียนฟังก์ชัน **`add_poly(p1, p2)`** และ **`mult_poly(p1, p2)`** ที่คืนผลบวก และผลคูณของ polynomial **`p1`** กับ **`p2`** ในโครงของโปรแกรมข้างล่างนี้

```
def add_poly(p1, p2):



def mult_poly(p1, p2):



for i in range(3):
    exec(input().strip())
```

ข้อมูลนำเข้า

คำสั่งภาษา Python ที่ใช้ทดสอบการทำงานของฟังก์ชัน

ข้อมูลส่งออก

ผลที่ได้จากการสั่งทำงานคำสั่งที่ได้รับ

ตัวอย่าง

input (จากแป้นพิมพ์)		output (ทางจอภาพ)	
`p1 = [(3,6),(2,4),(1,1),(-1,0)]` `p2 = [(3,4),(-1,1)]` `print(add_poly(p1, p2))`	$3x^6 + 2x^4 + x - 1$ $3x^4 - x$	`[(3, 6), (5, 4), (-1, 0)]`	$3x^6 + 5x^4 - 1$
`p1 = [(3,6),(2,4)]` `p2 = [(1,4),(-1,2)]` `print(mult_poly(p1, p2))`	$3x^6 + 2x^4$ $x^4 - x^2$	`[(3, 10), (-1, 8), (-2, 6)]`	$3x^{10} - x^8 - 2x^6$

10-09: ข้อมูลนิสิต

ข้อมูลของนิสิตวิศวฯ จุฬาฯ ประกอบด้วย ชื่อเล่น กรุ๊ป รุ่น และภาควิชา จงเขียนโปรแกรมเพื่อเลือกแสดงผลข้อมูลนิสิตที่อยู่ในกรุ๊ป รุ่น หรือภาควิชาที่ต้องการ โดยเรียงลาดับชื่อเล่นตามพจนานุกรม

ข้อมูลนำเข้า

บรรทัดแรก ระบุจานวนเต็ม n แทนจานวนข้อมูลนิสิต

n บรรทัดถัดมา ระบุข้อมูลนิสิตแต่ละคน ประกอบด้วย ชื่อเล่น กรุ๊ป รุ่น และภาควิชา (รับประกันว่า ชื่อเล่นจะไม่ซ้ำกัน)

บรรทัดสุดท้าย ระบุ กรุ๊ป รุ่น และ/หรือ ภาควิชา ที่จะใช้ค้นหาข้อมูลนิสิต (อาจระบุไม่ครบและไม่เรียงตามลาดับ)

ข้อมูลส่งออก

แสดงข้อมูลนิสิตที่อยู่ในกรุ๊ป รุ่น และภาควิชาที่กำหนด โดยเรียงลาดับชื่อเล่นตามพจนานุกรม ถ้าไม่มีให้แสดงว่า **`Not Found`**

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
8 Krit A 97 CP Oat A 98 CE Pim C 99 CP Pun C 97 CHE Jame Dog 100 CE Art C 97 CP Benz Dog 99 CP Mark C 100 CP CP C	Art C 97 CP Mark C 100 CP Pim C 99 CP
8 Krit A 97 CP Oat A 98 CE Pim C 99 CP Pun C 97 CHE Jame Dog 100 CE Art C 97 CP Benz Dog 99 CP Mark C 100 CP CP 97	Art C 97 CP Krit A 97 CP
8 Krit A 97 CP Oat A 98 CE Pim C 99 CP Pun C 97 CHE Jame Dog 100 CE Art C 97 CP Benz Dog 99 CP Mark C 100 CP 100	Jame Dog 100 CE Mark C 100 CP
8 Krit A 97 CP Oat A 98 CE Pim C 99 CP Pun C 97 CHE Jame Dog 100 CE Art C 97 CP Benz Dog 99 CP Mark C 100 CP 99 Dog CP	Benz Dog 99 CP

8 Krit A 97 CP Oat A 98 CE Pim C 99 CP Pun C 97 CHE Jame Dog 100 CE Art C 97 CP Benz Dog 99 CP Mark C 100 CP CE	Jame Dog 100 CE Oat A 98 CE
8 Krit A 97 CP Oat A 98 CE Pim C 99 CP Pun C 97 CHE Jame Dog 100 CE Art C 97 CP Benz Dog 99 CP Mark C 100 CP 99 CHE	Not Found

10-10: การเลือกภาควิชา

เมื่อนิสิตเรียนจบชั้นปีที่ 1 นิสิตจะต้องทำการเลือกภาควิชาที่จะเรียนต่อในชั้นปีที่ 2
โจทย์ข้อนี้จำลองวิธีที่ใช้ในการเลือกภาควิชาอย่างง่าย (ไม่ตรงกับวิธีที่ใช้จริง) โดยมีขั้นตอนดังนี้

1. แต่ละภาควิชาระบุจำนวนนิสิตที่สามารถรับได้
2. นิสิตแต่ละคน มีรหัสนิสิต คะแนนซึ่งคำนวณจากเกรดของทุกวิชาในชั้นปีที่ 1 และภาควิชาที่เลือก คนละ 4 ภาควิชา เรียงตามความต้องการลำดับที่ 1-4 (โจทย์ข้อนี้กำหนดให้ทุกภาควิชาใช้คะแนนเดียวกัน โดยไม่สนใจตัวคูณของแต่ละภาควิชา)
3. นิสิตที่มีคะแนนสูงสุดจะได้เลือกภาควิชาก่อน โดยจะได้เรียนในภาควิชาอันดับสูงที่สุดที่ยังมีที่ว่างอยู่ เช่น หากนิสิตเลือกภาค PE ME CP MT แต่ภาค PE และ ME รับนิสิตครบจำนวนแล้ว นิสิตจะได้เรียนในภาค CP
4. หลังจากกำหนดภาควิชาให้นิสิตครบแล้ว จะแสดงผลภาควิชาที่นิสิตแต่ละคนได้ เรียงตามรหัสนิสิต

ให้เขียนโปรแกรมเพื่อจำลองวิธีการที่ได้อธิบายข้างต้น

ข้อมูลนำเข้า

- บรรทัดแรก ระบุจำนวนเต็ม n แทนจำนวนภาควิชา
- n บรรทัดถัดมา ระบุชื่อภาควิชา และจำนวนนิสิตที่แต่ละภาควิชาสามารถรับได้ เป็นจำนวนเต็ม
- บรรทัดถัดมา ระบุจำนวนเต็ม m แทนจำนวนนิสิตที่ทำการเลือกภาควิชา
- m บรรทัดถัดมา ระบุข้อมูลนิสิตแต่ละคน ประกอบด้วยรหัสนิสิต 10 หลัก คะแนนเป็นจำนวนทศนิยม และภาควิชาที่นิสิตเลือก 4 ภาควิชา เรียงตามความต้องการลำดับที่ 1-4 (รับประกันว่าไม่มีนิสิตที่มีคะแนนเท่ากัน)

ข้อมูลส่งออก

แสดงภาควิชาที่นิสิตแต่ละคนได้ เรียงตามรหัสนิสิต (รับประกันว่านิสิตทุกคนจะมีภาควิชา)

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
5 CP 1 ME 2 PE 1 CHE 1 MT 3 6 59301234521 23.6 PE CP MT CHE 59300799921 44.5 ME CP CHE PE 59300081621 37 PE CHE MT CP 59300653521 61.2 PE MT CP ME 59300002121 19.4 CHE CP ME CP 59300048721 7 ME CP CHE MT	59300002121 ME 59300048721 MT 59300081621 CHE 59300653521 PE 59300799921 ME 59301234521 CP

10-11: รถไฟฟ้า

นักท่องเที่ยวชาวต่างชาติคนหนึ่งต้องการเดินทางท่องเที่ยวตามสถานที่สำคัญในกรุงเทพมหานคร เขาจึงเลือกใช้บริการรถไฟฟ้า เขาขึ้นรถไฟฟ้าที่สถานี A แต่เขามีเงินไม่มาก จึงเดินทางได้ไม่เกิน 2 ช่วงสถานีเท่านั้น เขาอยากทราบว่า จากสถานี A ที่เขาอยู่ตอนนี้ เขาจะสามารถไปถึงสถานีใดได้บ้าง

ข้อมูลนำเข้า

ส่วนแรกจะเป็นการบอกว่า สถานีรถไฟฟ้าคู่ใดอยู่ติดกันบ้าง แต่ละบรรทัดจะมีชื่อสถานี 2 ชื่อ คั่นด้วยเว้นวรรค หมายความว่า สถานี 2 สถานีนี้อยู่ติดกัน สถานีรถไฟฟ้าทั้งหมดไม่จำเป็นต้องเชื่อมต่อถึงกันก็ได้ (ลองดูในตัวอย่าง)

บรรทัดสุดท้ายมีชื่อสถานี 1 ชื่อ แทนสถานีที่นักท่องเที่ยวชาวต่างชาติอยู่ในตอนนี้ (ซึ่งอาจจะไม่เชื่อมกับสถานีใดเลยก็ได้)

ข้อมูลส่งออก

แสดงชื่อสถานีที่นักท่องเที่ยวชาวต่างชาติสามารถไปได้ (ห่างกันไม่เกิน 2 ช่วงสถานี) ให้ตอบเรียงตามลำดับพจนานุกรม

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
Siam ChitLom ChitLom PhloenChit PhloenChit Nana Siam NationalStadium Ratchadamri Siam Siam PhayaThai Ratchadamri SalaDaeng ThongLo Ekkamai Ekkamai ThongLo Siam	ChitLom NationalStadium PhayaThai PhloenChit Ratchadamri SalaDaeng Siam
Siam ChitLom ChitLom PhloenChit PhloenChit Nana Siam NationalStadium Ratchadamri Siam Siam PhayaThai Ratchadamri SalaDaeng ThongLo Ekkamai Ekkamai ThongLo ThongLo	Ekkamai ThongLo
Siam ChitLom ChitLom PhloenChit PhloenChit Nana Siam NationalStadium Ratchadamri Siam Siam PhayaThai Ratchadamri SalaDaeng ThongLo Ekkamai Ekkamai ThongLo MoChit	MoChit

11: NumPy

https://numpy.org/

11-01: ฟังก์ชันเกี่ยวกับ Indexing & Slicing

จงเขียนฟังก์ชันที่ทำงานตามชื่อฟังก์ชัน (หรือตามที่เขียนใน comment)

```
import numpy as np

# A is a 2-d array
def get_column_from_bottom_to_top( A, c ):


def get_odd_rows( A ):


def get_even_column_last_row( A ):


def get_diagonal1( A ): # A is a square matrix
    # from top-left corner down to bottom-right corner


def get_diagonal2( A ): # A is a square matrix
    # from top-right corner down to bottom-left corner


exec(input().strip())
```

ข้อมูลนำเข้า

คำสั่งภาษา Python ที่ใช้ทดสอบการทำงานของฟังก์ชัน

ข้อมูลส่งออก

ผลที่ได้จากการสั่งทำงานคำสั่งที่ได้รับ

ตัวอย่าง

input (จากแป้นพิมพ์)	output
`A=np.array([[1,2],[3,4]]);print(get_column_from_bottom_to_top(A,1))`	`[4 2]`
`A=np.array([[1,2],[3,4],[5,6],[7,8]]);print(get_odd_rows(A))`	`[[3 4]` ` [7 8]]`
`A=np.array([[1,2,3],[4,5,6]]);print(get_even_column_last_row(A))`	`[4 6]`
`A=np.array([[1,2,3],[4,5,6],[7,8,9]]);print(get_diagonal1(A))`	`[1 5 9]`
`A=np.array([[1,2,3],[4,5,6],[7,8,9]]);print(get_diagonal2(A))`	`[3 5 7]`

11-02: ฟังก์ชันเกี่ยวกับการคำนวณอาเรย์กับค่าสเกล่าร์

จงเขียนฟังก์ชันที่ทำงานตามชื่อฟังก์ชัน (หรือตามที่เขียนใน comment)

```
import numpy as np

def toCelsius( f ):
    # f เป็นอาเรย์หนึ่งมิติเก็บอุณหภูมิในหน่วยองศาฟาเรนไฮต์
    # คืนอาเรย์หนึ่งมิติที่เก็บอุณหภูมิในหน่วยองศาเซลเซียสที่ได้จากการแปลงแต่ละอุณหภูมิใน f



def BMI( wh ):
    # wh เป็นอาเรย์สองมิติขนาด nx2 แทนน้ำหนัก (หน่วยเป็น กก.) และความสูง (หน่วยเป็น ซม.)
    # ของคน n คน คอลัมน์ 0 เก็บน้ำหนัก คอลัมน์ 1 เก็บความสูง [[w1,h1], [w2,h2], ...]
    # คืนอาเรย์หนึ่งมิติที่เก็บค่า body mass index ของทุกคนใน wh

                                        BMI = weight_(in Kg.) / height^2_(in m.)

def distanceTo( p, Points ):
    # p เป็นอาเรย์หนึ่งมิติขนาด 2 ช่องแทนจุดหนึ่งจุด ช่อง 0 เก็บพิกัด x ช่อง 1 เก็บพิกัด y
    # Points เป็นอาเรย์สองมิติขนาด nx2 เก็บพิกัดของจุดจำนวน n จุด
    # คืนอาเรย์หนึ่งมิต n ช่อง ที่เก็บระยะทางที่วัดจากจุด p ถึงแต่ละจุดใน Points



exec(input().strip())
```

ข้อมูลนำเข้า

คำสั่งภาษา Python ที่ใช้ทดสอบการทำงานของฟังก์ชัน

ข้อมูลส่งออก

ผลที่ได้จากการสั่งทำงานคำสั่งที่ได้รับ

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
`print(toCelsius(np.array([32,212])))`	`[  0. 100.]`
`print(BMI(np.array([[60,170],[50,160]])))`	`[20.76124567` `19.53125    ]`
`print(distanceTo([0,0],np.array([[3,0],[0,4],[3,4]])))`	`[3. 4. 5.]`

11-03: ฟังก์ชันการทำนายผลการเรียน

นักวิจัยรายหนึ่งสร้างสูตรทำนายโอกาส $p(x)$ ที่นักเรียน x จะเรียนผ่านวิชาหนึ่งจากจำนวนโจทย์ที่ทำ (x_0) กับเกรดเฉลี่ยที่มี (x_1) ข้างล่างนี้

$$p(x) = \frac{1}{1 + e^{-logit(x)}}$$

$$logit(x) = -3.98 + 0.1x_0 + 0.5x_1$$

จงเขียนฟังก์ชัน **p(X)** ที่ทำงานตามที่เขียนใน comment

```
import numpy as np

def p( X ):
    # X เป็นอาเรย์ขนาด nx2 เก็บจำนวนโจทย์ที่ทำ (คอลัมน์ 0) กับเกรดเฉลี่ย (คอลัมน์ 1) ของนักเรียน n คน
    # คืนอาเรย์ขนาด n ช่อง เก็บความน่าจะเป็นที่นักเรียนแต่ละคนจะเรียนผ่านวิชา คำนวณจากสูตรข้างบน
    # ใช้ความสามารถของ NumPy จะเขียนได้โดยไม่ต้องใช้วงวน (อย่างมาก 3 บรรทัด)



exec(input().strip())
```

ข้อมูลนำเข้า

คำสั่งภาษา Python ที่ใช้ทดสอบการทำงานของฟังก์ชัน

ข้อมูลส่งออก

ผลที่ได้จากการสั่งทำงานคำสั่งที่ได้รับ

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
`print(p(np.array([[100, 4.00]])))`	`[0.99967129]`
`print(p(np.array([[80, 2.50], [1, 4.00]])))`	`[0.99488271 0.13238887]`

11-04: ฟังก์ชันเกี่ยวกับ slicing & element-wise operation

จงเขียนฟังก์ชันที่ทำงานตามชื่อฟังก์ชัน (หรือตามที่เขียนใน comment)

```
import numpy as np

def sum_2_rows( M ):
    # คืนผลที่ได้จากการรวมจำนวนในคอลัมน์เดียวกันของแถวที่ติดกันทีละคู่แถว
    # เช่น M = [[ 0,  1,  2,  3],   ได้  [[ 4,  6,  8, 10],
    #          [ 4,  5,  6,  7],         [20, 22, 24, 26]]
    #          [ 8,  9, 10, 11],
    #          [12, 13, 14, 15]]



def sum_left_right( M ):
    # คืนผลที่ได้จากการรวมจำนวนของครึ่งซ้ายกับครึ่งขวาของ M
    # เช่น M = [[ 0,  1,  2,  3],   ได้  [[ 2,  4],
    #          [ 4,  5,  6,  7],         [10, 12],
    #          [ 8,  9, 10, 11],         [18, 20],
    #          [12, 13, 14, 15]]         [26, 28]]



def sum_upper_lower( M ):
    # คืนผลที่ได้จากการรวมจำนวนของครึ่งบนกับครึ่งล่างของ M
    # เช่น M = [[ 0,  1,  2,  3],   ได้  [[ 8, 10, 12, 14],
    #          [ 4,  5,  6,  7],         [16, 18, 20, 22]]
    #          [ 8,  9, 10, 11],
    #          [12, 13, 14, 15]]



def sum_4_quadrants( M ):
    # คืนผลที่ได้จากการแบ่ง M เป็น 4 จตุภาค และรวมจำนวนที่ตำแหน่งตรงกันในแต่ละจตุภาค
    # เช่น M = [[ 0,  1,  2,  3],   ได้  [[20, 24],
    #          [ 4,  5,  6,  7],         [36, 40]]
    #          [ 8,  9, 10, 11],
    #          [12, 13, 14, 15]]



def sum_4_cells( M ):
    # คืนผลที่ได้จากการรวมจำนวนที่ติดกัน 4 ตัว ตามรูปแบบในตัวอย่างข้างล่างนี้
    # เช่น M = [[ 0,  1,  2,  3],   ได้  [[10, 18],
    #          [ 4,  5,  6,  7],         [42, 50]]
    #          [ 8,  9, 10, 11],
    #          [12, 13, 14, 15]]



def count_leap_years( years ):
    # years เป็นอาเรย์เก็บปี พ.ศ.
    # คืนจำนวนปีใน years ที่เป็นปีอธิกสุรทิน (ปีที่ ก.พ. มี 29 วัน)



exec(input().strip())
```

ข้อมูลนำเข้า

คำสั่งภาษา Python ที่ใช้ทดสอบการทำงานของฟังก์ชัน

ข้อมูลส่งออก

ผลที่ได้จากการสั่งทำงานคำสั่งที่ได้รับ

ตัวอย่าง

คำสั่ง **`np.arange(36).reshape(6,6)`** ได้อาเรย์

```
[[ 0  1  2  3  4  5]
 [ 6  7  8  9 10 11]
 [12 13 14 15 16 17]
 [18 19 20 21 22 23]
 [24 25 26 27 28 29]
 [30 31 32 33 34 35]]
```

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
`print(sum_2_rows(np.arange(36).reshape(6,6)))`	`[[ 6  8 10 12 14 16]` ` [30 32 34 36 38 40]` ` [54 56 58 60 62 64]]`
`print(sum_left_right(np.arange(36).reshape(6,6)))`	`[[ 3  5  7]` ` [15 17 19]` ` [27 29 31]` ` [39 41 43]` ` [51 53 55]` ` [63 65 67]]`
`print(sum_upper_lower(np.arange(36).reshape(6,6)))`	`[[18 20 22 24 26 28]` ` [30 32 34 36 38 40]` ` [42 44 46 48 50 52]]`
`print(sum_4_quadrants(np.arange(36).reshape(6,6)))`	`[[42 46 50]` ` [66 70 74]` ` [90 94 98]]`
`print(sum_4_cells(np.arange(36).reshape(6,6)))`	`[[ 14  22  30]` ` [ 62  70  78]` ` [110 118 126]]`
`print(count_leap_years(np.array([2543,2559,2560])))`	`2`

11-05: ฟังก์ชันผลิตอาเรย์สูตรคูณ

จงเขียนฟังก์ชันที่ทำงานตามชื่อฟังก์ชัน (หรือตามที่เขียนใน comment)

```
import numpy as np

def mult_table(nrows, ncols):
    # คืนอาเรย์ที่มี shape เป็น (nrow, ncols) ภายในเก็บตารางสูตรคูณ (ดูตัวอย่างข้างล่าง)



exec(input().strip())
```

ข้อแนะนำ: ถ้าคิดไม่ออก ลองอ่านการคำนวณ outer product จาก https://en.wikipedia.org/wiki/Outer_product

ข้อมูลนำเข้า

คำสั่งภาษา Python ที่ใช้ทดสอบการทำงานของฟังก์ชัน

ข้อมูลส่งออก

ผลที่ได้จากการสั่งทำงานคำสั่งที่ได้รับ

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
`print(mult_table(2,2))`	`[[1 2]` ` [2 4]]`
`print(mult_table(3,4))`	`[[ 1  2  3  4]` ` [ 2  4  6  8]` ` [ 3  6  9 12]]`
`print(mult_table(12,12))`	`[[  1   2   3   4   5   6   7   8   9  10  11  12]` ` [  2   4   6   8  10  12  14  16  18  20  22  24]` ` [  3   6   9  12  15  18  21  24  27  30  33  36]` ` [  4   8  12  16  20  24  28  32  36  40  44  48]` ` [  5  10  15  20  25  30  35  40  45  50  55  60]` ` [  6  12  18  24  30  36  42  48  54  60  66  72]` ` [  7  14  21  28  35  42  49  56  63  70  77  84]` ` [  8  16  24  32  40  48  56  64  72  80  88  96]` ` [  9  18  27  36  45  54  63  72  81  90  99 108]` ` [ 10  20  30  40  50  60  70  80  90 100 110 120]` ` [ 11  22  33  44  55  66  77  88  99 110 121 132]` ` [ 12  24  36  48  60  72  84  96 108 120 132 144]]`

11-06: ใครได้คะแนนรวมน้อยกว่าคะแนนเฉลี่ย

จงเขียนฟังก์ชันที่ทำงานตามชื่อฟังก์ชัน (และตามที่เขียนใน comment)

```
import numpy as np

def read_data():
    # อ่านข้อมูลจากแป้นพิมพ์ จากนั้นสร้างและคืนอาเรย์สองตัว
    # weight เป็นอาเรย์สามช่องเก็บน้ำหนักของคะแนนกลางภาค ปลายภาค และโครงงาน (float)
    # data เป็นอาเรย์ขนาด nx4 เก็บข้อมูลนักเรียน n คน แต่ละคนมีข้อมูล
    #       เลขประจำตัว คะแนนกลางภาค ปลายภาค และโครงงาน (int)

    w = [float(e) for e in input().split()]
    weight = np.array(w)
    n = int(input())
    data = np.ndarray((n, 4), int)
    for i in range(n):
        data[i] = [int(e) for e in input().split()]
    return weight, data

def report_lower_than_mean(weight, data):
    # แสดงเลขประจำตัวที่ได้คะแนนรวมต่ำกว่าคะแนนเฉลี่ย
    #  - คะแนนรวม คำนวณมาจากผลรวมของ คะแนนแต่ละส่วนคูณด้วยน้ำหนักของแต่ละส่วน
    #  - คะแนนเฉลี่ย คือค่าเฉลี่ยของคะแนนรวมต่าง ๆ
    # ให้แสดงบนบรรทัดเดียวกันหมดคั่นด้วยเครื่องหมายจุลภาคและช่องว่างหนึ่งช่อง
    # เรียงตามลำดับที่ปรากฏใน data  ถ้าไม่มีใครได้ต่ำกว่าคะแนนเฉลี่ยเลย ให้แสดงคำว่า None



exec(input().strip())
```

ข้อแนะนำ: ถ้าคิดไม่ออก ลองอ่านการคำนวณ outer product จาก https://en.wikipedia.org/wiki/Outer_product

ข้อมูลนำเข้า

คำสั่งภาษา Python ที่ใช้ทดสอบการทำงานของฟังก์ชัน

ข้อมูลส่งออก

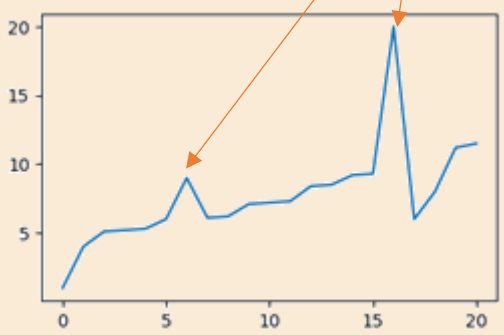
ผลที่ได้จากการสั่งทำงานคำสั่งที่ได้รับ

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
`w,d = read_data(); report_lower_than_mean(w,d)` `0.3 0.5 0.2` `5` `610111 80 90 70` `610222 50 80 68` `610333 70 85 80` `610444 60 50 90` `610555 90 74 70`	`610222, 610444`
`w,d = read_data(); report_lower_than_mean(w,d)` `0.3 0.5 0.2` `2` `610111 80 90 80` `610222 90 80 90`	`None`

11-07: การหาตำแหน่งของยอด

ถ้านำข้อมูลในลิสต์ y = [1, 4, 5.1, 5.2, 5.3, 6, 9, 6.1, 6.2, 7.1, 7.2, 7.3, 8.4, 8.5, 9.2, 9.3, 20, 6, 8, 11.2, 11.5] ไปวาดกราฟเส้นจะได้ดังรูปข้างล่างนี้ จะเห็นว่ามีการเปลี่ยนแปลงของข้อมูลที่มองได้ว่าเป็น "ยอด" คือ ข้อมูลตัวที่เป็นยอด มีค่ามากกว่าทั้งตัวติดกันทางด้านซ้ายและด้านขวา จงเขียนคำสั่งในฟังก์ชัน peak_indexes ของโปรแกรมข้างล่างนี้ ที่รับรายการของจำนวน แล้วแสดงตำแหน่งทั้งหมดที่เป็นยอด




```
import numpy as np

def peak_indexes(x):
    # x เป็นอาเรย์เก็บจำนวนต่าง ๆ
    # คืนอาเรย์ที่เก็บตำแหน่งใน x ที่เป็น "ยอด"

    ???


def main():
    d = np.array([float(e) for e in input().split()])
    pos = peak_indexes(np.array(d))
    if len(pos) > 0:
        print(", ".join([str(e) for e in pos]))
    else:
        print("No peaks")

exec(input().strip())    # Don't remove this line
```

ฝึกเขียนด้วยคำสั่ง NumPy โดยไม่ต้องใช้คำสั่ง loop เลย

ข้อมูลนำเข้า

รายการของจำนวนจริงบนบรรทัดเดียวกันแต่ละตัวคั่นด้วยช่องว่าง

ข้อมูลส่งออก

รายการของตำแหน่งทั้งหมดของยอดคั่นด้วยจุลภาคและช่องว่างหนึ่งช่อง ถ้าไม่มียอดให้แสดง **No peaks**

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
`print(peak_indexes(np.array([1,2,3,4])))`	`[]`
`main()` `1 2 3 4`	`No peaks`
`main()` `1 9 1 9 1 9 1 9 1 9 1`	`1, 3, 5, 7, 9`

12: Class & Object

https://pixabay.com/photos/tray-breakfast-muesli-fruits-bowls-2546077/

12-01: จำนวนเชิงซ้อน

ให้นิสิตสร้างคลาส **Complex** ซึ่งรองรับการทำงานของจำนวนเชิงซ้อน **a+bi** เมื่อ **a** คือส่วนจริงและ **b** คือส่วนจินตภาพ โจทย์ข้อนี้ให้เขียนคลาส **Complex** ที่มีโครงและตัวอย่างการใช้งานดังนี้

โครงของคลาส Complex	ตัวอย่างการใช้งาน Complex
`class Complex :` `  def __init__(self,a,b):` `  def __str__(self):` `  def __add__(self, rhs):` `  def __mul__(self, rhs):` `  def __truediv__(self, rhs):`	`a = Complex(3,4)` `b = Complex(5,6)` `c = Complex(3,1)` `d = Complex(2,1)` `print( str(a) )  # ได้ 3+4i` `print( a+b )  # ได้ 8+10i` `print( a*b )  # ได้ -9+38i` `print( b*a )  # ได้ -9+38i` `print( c/d )  # ได้ 1.4-0.2i`

เมท็อด **__str__** ค่อนข้างซับซ้อน เพราะ โจทย์ข้อนี้ต้องการผลลัพธ์ที่เหมือนปกติมากที่สุด เช่น **print(Complex(2,0))** ต้องได้ผลลัพธ์เป็น **2** ไม่ใช่ **2+0i** หรือ **print(Complex(2,-1))** ต้องได้ผลลัพธ์เป็น **2-i** ไม่ใช่ **2-1i** ให้ดูกรณีต่าง ๆ ตามตัวอย่างในหน้าถัดไป

เมท็อด **__add__** ถูกเรียกเมื่อเราใช้ตัวปฏิบัติการ **+** กับ **Complex** สองตัว ได้ผลลัพธ์เป็น **Complex** ใหม่ที่แทนผลบวกที่ได้

เมท็อด **__mul__** ถูกเรียกเมื่อเราใช้ตัวปฏิบัติการ ***** กับ **Complex** สองตัว ได้ผลลัพธ์เป็น **Complex** ใหม่ที่แทนผลคูณที่ได้

$$(a+bi)*(c+di) = (ac-bd)+(ad+bc)i$$

เมท็อด **__truediv__** ถูกเรียกเมื่อเราใช้ตัวปฏิบัติการ **/** กับ **Complex** สองตัว ได้ผลลัพธ์เป็น **Complex** ใหม่ที่แทนผลหารที่ได้

$$\frac{a+bi}{c+di} = \frac{(a+bi)(c-di)}{(c+di)(c-di)} = \frac{(ac+bd)+(-ad+bc)i}{c^2+d^2} = \frac{ac+bd}{c^2+d^2} + \frac{-ad+bc}{c^2+d^2}i$$

การส่งตรวจ

ให้นำโปรแกรมข้างล่างนี้ ต่อท้าย **class Complex** ที่เขียนข้างบนนี้ แล้วจึงส่งให้ Grader ตรวจ

```
t, a, b, c, d = [int(x) for x in input().split()]
c1 = Complex(a,b)
c2 = Complex(c,d)
if   t == 1 : print(c1)
elif t == 2 : print(c2)
elif t == 3 : print(c1+c2)
elif t == 4 : print(c1*c2)
else        : print(c1/c2)
```

ข้อมูลนำเข้า

จำนวนเต็ม 5 ตัว คั่นด้วยช่องว่าง (ดูตัวอย่าง และโปรแกรมที่ส่งตรวจประกอบ)

ข้อมูลส่งออก

ผลการทำงานของโปรแกรมข้างบนที่อาศัยคลาส **Complex** ที่เขียน

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
`1 3 4 5 6`	`3+4i`
`2 3 4 5 6`	`5+6i`
`1 0 3 3 0`	`3i`
`2 0 3 3 0`	`3`
`1 -3 3 3 -3`	`-3+3i`
`2 -3 3 3 -3`	`3-3i`
`1 -3 -3 0 -3`	`-3-3i`
`2 -3 -3 0 -3`	`-3i`
`1 3 1 3 1`	`3+i`
`1 3 -1 3 1`	`3-i`
`1 0 1 0 -1`	`i`
`2 0 1 0 -1`	`-i`
`3 3 4 5 6`	`8+10i`
`4 3 1 2 1`	`-9+38i`
`5 3 1 2 1`	`1.4-0.2i`

12-02: คลาสของไพ่

ข้างล่างนี้แสดงการเรียกใช้คลาส **Card** ซึ่งแทนไพ่ 1 ใบ ประกอบด้วย ค่าของไพ่ (**value**) ซึ่งเป็นสตริง **"A", "2", "3", … , "10", "J", "Q", "K"** และ ดอกของไพ่ (**suit**) ซึ่งเป็นสตริงเช่นกัน **"club", "diamond", "heart", "spade"** โปรแกรมข้างล่างนี้รับไพ่เข้ามาหลายใบมาสร้างเป็นลิสต์ของไพ่ (**cards**) และมีการเรียกใช้เมท็อดต่าง ๆ ของคลาส **Card** ให้นิสิตเขียนเมท็อดต่าง ๆ ของคลาส **Card** ให้สมบูรณ์ (ห้ามแก้ไขบริเวณที่มีพื้นหลังสีเทา)

```
class Card:
    def __init__(self, value, suit):
        ???

    def __str__(self):
        ???

    def getScore(self):
        ???

    def sum(self, other):
        ???

    def __lt__(self, rhs):
        ???

n = int(input())
cards = []
for i in range(n):
    value, suit = input().split()
    cards.append(Card(value, suit))
for i in range(n):
    print(cards[i].getScore())
print("----------")
for i in range(n-1):
    print(Card.sum(cards[i], cards[i+1]))
print("----------")
cards.sort()
for i in range(n):
    print(cards[i])
```

รายละเอียดต่าง ๆ ของคลาส **Card** และเมท็อดของคลาส **Card**

- เมท็อด **getScore** จะคืนค่าคะแนนของไพ่เป็นจานวนเต็ม ตามกฎดังนี้
  - ไพ่ที่มีค่า **A** จะมีคะแนน 1 คะแนน
  - ไพ่ที่มีค่า **2** ถึง **10** จะมีคะแนนเท่ากับค่าของไพ่ คือ 2 ถึง 10 คะแนน ตามลำดับ
  - ไพ่ที่มีค่า **J, Q, K** จะมีคะแนน 10 คะแนน
- เมท็อด **sum** จะคืนค่าผลรวมคะแนนของไพ่สองใบและ **mod** ด้วย 10 เช่น
  - **Card.sum(Card("7", "club"), Card("2", "heart"))** ได้ผลลัพธ์เป็น 9
  - **Card.sum(Card("J", "spade"), Card("5", "diamond"))** ได้ผลลัพธ์เป็น 5
- การเรียงลำดับของไพ่เป็นดังนี้
  - ค่าของไพ่เรียงตามลำดับดังนี้ **3 < 4 < 5 < … < 10 < J < Q < K < A < 2**
  - ดอกของไพ่เรียงตามลำดับดังนี้ **club < diamond < heart < spade**
  - ถ้าไพ่สองใบมีค่าไม่เท่ากัน ไพ่ที่มีค่ามากกว่าจะเป็นไพ่ที่มากกว่า
  - ถ้าไพ่สองใบมีค่าเท่ากัน ไพ่ที่มีดอกสูงกว่าจะเป็นไพ่ที่มากกว่า

ข้อมูลนำเข้า

บรรทัดแรกมีจำนวนเต็ม n แทนจำนวนไพ่ที่จะตามมา

n บรรทัดต่อมา แต่ละบรรทัดมีค่าและดอกของไพ่แต่ละใบ คั่นด้วยช่องว่าง

ข้อมูลส่งออก

มี $3n$ +1 บรรทัด

n บรรทัดแรก แสดงคะแนนของไพ่แต่ละใบ ตามด้วยขีดคั่น 1 บรรทัด

n -1 บรรทัดถัดมา แสดงคะแนนรวมของไพ่ 2 ใบที่ติดกันในลำดับ ตามด้วยขีดคั่น 1 บรรทัด

n บรรทัดสุดท้าย แสดงไพ่เรียงตามลำดับ

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
5 A spade K heart K club 7 diamond 2 spade	1 10 10 7 2 ---------- 1 0 7 9 ---------- (7 diamond) (K club) (K heart) (A spade) (2 spade)

12-03: ไพ่ใบถัดไป

ข้างล่างนี้แสดงการเรียกใช้คลาส **Card** ซึ่งแทนไพ่ 1 ใบ ประกอบด้วย ค่าของไพ่ (**value**) ซึ่งเป็นสตริง **"A", "2", "3", … , "10", "J", "Q", "K"** และ ดอกของไพ่ (**suit**) ซึ่งเป็นสตริงเช่นกัน **"club", "diamond", "heart", "spade"** โปรแกรมข้างล่างนี้รับไพ่เข้ามาหลายใบมาสร้างเป็นลิสต์ของไพ่ (**cards**) และมีการเรียกใช้เมท็อดต่าง ๆ ของคลาส **Card** ให้นิสิตเขียนเมท็อดต่าง ๆ ของคลาส **Card** ให้สมบูรณ์ (ห้ามแก้ไขบริเวณที่มีพื้นหลังสีเทา)

```
class Card:
    def __init__(self, value, suit):
        ???

    def __str__(self):
        ???

    def next1(self):
        ???

    def next2(self, other):
        ???

n = int(input())
cards = []
for i in range(n):
    value, suit = input().split()
    cards.append(Card(value, suit))
for i in range(n):
    print(cards[i].next1())
print("----------")
for i in range(n):
    print(cards[i])
print("----------")
for i in range(n):
    cards[i].next2()
    cards[i].next2()
    print(cards[i])
```

รายละเอียดต่าง ๆ ของคลาส **Card** และเมท็อดของคลาส **Card**

- การเรียงลำดับของไพ่เป็นดังนี้
  - ค่าของไพ่เรียงตามลำดับดังนี้ **3 < 4 < 5 < … < 10 < J < Q < K < A < 2**
  - ดอกของไพ่เรียงตามลำดับดังนี้ **club < diamond < heart < spade**
  - ถ้าไพ่สองใบมีค่าไม่เท่ากัน ไพ่ที่มีค่ามากกว่าจะเป็นไพ่ที่มากกว่า
  - ถ้าไพ่สองใบมีค่าเท่ากัน ไพ่ที่มีดอกสูงกว่าจะเป็นไพ่ที่มากกว่า
- ไพ่ใบถัดไป คือไพ่ที่มีค่าสูงกว่าไพ่ที่สนใจอยู่ 1 ลำดับ ยกเว้นไพ่ใบถัดไปของไพ่ที่มีค่าสูงที่สุด (**2 spade**) จะเป็นไพ่ใบที่ต่ำที่สุด (**3 club**) เช่น
  - ไพ่ใบถัดไปของ (**5 diamond**) คือ (**5 heart**)
  - ไพ่ใบถัดไปของ (**10 spade**) คือ (**J club**)
- เมท็อด **next1** จะคืนค่าออบเจ็กต์ใหม่ซึ่งเป็นไพ่ใบถัดไป
- เมท็อด **next2** จะแก้ไขค่าของออบเจ็กต์ให้เป็นไพ่ใบถัดไป โดยไม่มีการสร้างออบเจ็กต์ใหม่
- การเรียกใช้เมท็อด **next2** สองครั้ง จะทาให้ไพ่ถูกแก้ไขเป็นไพ่ที่อยู่ถัดไป 2 ลำดับ

ข้อมูลนำเข้า

บรรทัดแรกมีจำนวนเต็ม n แทนจำนวนไพ่ที่จะตามมา

n บรรทัดต่อมา แต่ละบรรทัดมีค่าและดอกของไพ่แต่ละใบ คั่นด้วยช่องว่าง

ข้อมูลส่งออก

มี $3n+2$ บรรทัด

n บรรทัดแรก แสดงไพ่ใบถัดไปซึ่งเป็นผลลัพธ์ของเมท็อด **`next1`** ตามด้วยขีดคั่น 1 บรรทัด

n -1 บรรทัดถัดมา แสดงไพ่แต่ละใบในลิสต์ **`cards`** ตามด้วยขีดคั่น 1 บรรทัด

n บรรทัดสุดท้าย แสดงไพ่แต่ละใบหลังการเรียกใช้เมท็อด **`next2`** สองครั้ง

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
5 A spade K heart K club 7 diamond 2 spade	(2 club) (K spade) (K diamond) (7 heart) (3 club) ---------- (A spade) (K heart) (K club) (7 diamond) (2 spade) ---------- (2 diamond) (A club) (K heart) (7 spade) (3 diamond)

12-04: จุดในสี่เหลี่ยมผืนผ้า

ข้างล่างนี้แสดงคลาส **Point** แทนจุดในระนาบสองมิติ และมีคลาส **Rect** แทนสี่เหลี่ยมผืนผ้าที่ภายในเก็บจุดที่มุมซ้ายล่างกับมุมขวาบนของสี่เหลี่ยมผืนผ้า (มีด้านที่ ขนานกับแกน x หรือแกน y) สิ่งที่ต้องการให้เขียนคือ เมท็อด **area** และ **contains** ของคลาส **Rect** เมท็อด **area** คืนพื้นที่ของสี่เหลี่ยม ส่วน **contains** ทดสอบว่าจุดที่ได้รับอยู่ภายในสี่เหลี่ยมหรือไม่ (อยู่ที่ขอบสี่เหลี่ยมก็ถือว่าอยู่ในสี่เหลี่ยม) ถ้าอยู่ภายในคืน **True** ถ้าอยู่ข้างนอกก็คืน **False**

```
class Point:
    def __init__(self, x, y):
        self.x = x
        self.y = y
    def __str__(self):
        return "("+str(self.x)+","+str(self.y)+")"

class Rect:
    def __init__(self, p1, p2):
        self.lowerleft = p1
        self.upperright = p2

    def area(self):
        ???

    def contains(self, p):
        ???

x1,y1,x2,y2 = [int(e) for e in input().split()]
lowerleft = Point(x1,y1)
upperright = Point(x2,y2)
rect = Rect(lowerleft, upperright)
print(rect.area())
m = int(input())
for i in range(m):
    x,y = [int(e) for e in input().split()]
    p = Point(x,y)
    print(rect.contains(p))
```

โปรแกรมในบริเวณสีเทามีไว้อ่านจุดสองจุดของมุมสี่เหลี่ยม จากนั้นแสดงพื้นที่ แล้วก็รับจุดจำนวนหนึ่งมาเพื่อทดสอบว่าอยู่ภายในสี่เหลี่ยมหรือไม่ บริเวณสีเทานี้ไม่ต้องแก้ไขใด นิสิตเขียนคำสั่งเฉพาะในเมท็อด **area** และ **contains** ก็พอ

ข้อมูลนำเข้า

บรรทัดแรกมีจำนวนเต็มสี่จำนวนแทนพิกัด x, y ของมุมซ้ายล่างกับขวาบนของสี่เหลี่ยม

บรรทัดต่อมามีจำนวนเต็ม m แทนจำนวนกรณีทดสอบที่จะตามมา

m บรรทัดต่อมา แต่ละบรรทัดมีจำนวนเต็ม 2 จำนวนแทนพิกัด x, y ที่จะนำไปทดสอบว่า อยู่ในสี่เหลี่ยมที่รับตอนแรกหรือไม่

ข้อมูลส่งออก

บรรทัดแรกแสดงพื้นที่ของสี่เหลี่ยม ตามด้วยผลการทดสอบอีก m บรรทัด

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
2 2 10 10 4 0 0 2 4 3 5 10 1	64 False True True False

12-05: การเรียงลำดับสี่เหลี่ยมผืนผ้าตามพื้นที่

ข้างล่างนี้แสดงคลาส **Point** แทนจุดในระนาบสองมิติ และมีคลาส **Rect** แทนสี่เหลี่ยมผืนผ้าที่ภายในเก็บจุดที่มุมซ้ายล่างกับมุมขวาบนของสี่เหลี่ยมผืนผ้า (มีด้านที่ ขนานกับแกน x หรือแกน y) โปรแกรมข้างล่างนี้อ่านสี่เหลี่ยมต่าง ๆ เข้ามา เรียงลำดับตามพื้นที่จากน้อยไปมาก แล้วแสดงผลทางจอภาพ

```
class Point:
    def __init__(self, x, y):
        self.x = x
        self.y = y
    def __str__(self):
        return "("+str(self.x)+","+str(self.y)+")"

class Rect:
    def __init__(self, p1, p2):
        self.lowerleft = p1
        self.upperright = p2

    def __str__(self):
        return str(self.lowerleft)+"-"+str(self.upperright)

    ???


n = int(input())
rects = []
for i in range(n):
    x1,y1,x2,y2 = [int(e) for e in input().split()]
    rects.append(Rect(Point(x1,y1), Point(x2,y2)))
rects.sort()
for i in range(n):
    print(rects[i])
```

จงปรับปรุงคลาส **Rect** ให้สามารถเปรียบเทียบกันได้ว่าใครน้อยกว่าใคร โดยใช้พื้นที่เป็นตัวเปรียบเทียบ (เช่น **r1** และ **r2** เก็บสี่เหลี่ยม สามารถเขียน **r1 < r2** เพื่อเปรียบเทียบได้) เมื่อทำได้เช่นนี้ จะสามารถ sort list of **Rect** objects ได้ง่าย ๆ ตามโปรแกรมที่เขียนข้างบนนี้ (ห้ามแก้ไขบริเวณที่มีพื้นหลังสีเทา)

ข้อมูลนำเข้า

บรรทัดแรกมีจำนวนเต็ม n แทนจำนวนlเหลี่ยมกรณีทดสอบที่จะตามมา

n บรรทัดต่อมา แต่ละบรรทัดมีจำนวนเต็มสี่จำนวนแทนพิกัด x, y ของมุมซ้ายล่างกับขวาบนของสี่เหลี่ยม

ข้อมูลส่งออก

ลำดับของสี่เหลี่ยมเรียงตามพื้นที่จากน้อยไปมาก

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
3 1 1 3 3 0 0 10 10 2 2 3 3	(2,2)-(3,3) (1,1)-(3,3) (0,0)-(10,10)

12-06: กระปุกออมสิน 1

โจทย์ข้อนี้ให้สร้างคลาส **piggybank** เพื่อผลิตอ็อบเจกต์กระปุกออมสิน ที่สามารถหยอดเหรียญ 1, 2, 5, และ 10 ได้ไม่จำกัด ผ่านเมท็อด **add1**, **add2**, **add5**, และ **add10** สามารถใช้ **int( )** ที่คืนมูลค่ารวมของเงินในกระปุก และสามารถเปรียบเทียบมูลค่าสองกระปุกด้วย **<** ได้ ตามโครงของคลาสและตัวอย่างการใช้งานข้างล่างนี้ จงเขียนคลาสนี้ให้สมบูรณ์

โครงของคลาส piggybank

```
class piggybank:
  def __init__(self):
     # มีตัวแปร 4 ตัวเก็บจำนวนเหรียญของเหรียญแต่ละแบบ

  def add1(self, n):
     # เพิ่ม n ในตัวแปรที่เก็บจำนวนเหรียญบาท

  def add2(self, n):
     # เพิ่ม n ในตัวแปรที่เก็บจำนวนเหรียญสองบาท

  def add5(self, n):
     # เพิ่ม n ในตัวแปรที่เก็บจำนวนเหรียญห้าบาท

  def add10(self, n):
     # เพิ่ม n ในตัวแปรที่เก็บจำนวนเหรียญสิบบาท

  def __int__(self):
     # คืนมูลค่ารวม = ค่าของเหรียญคูณกับจำนวนเหรียญ

  def __lt__(self, rhs):
     # เปรียบเทียบจำนวนเงินใน self กับจำนวนเงินใน rhs

  def __str__(self):
     # คืนสตริงที่แสดงจำนวนเหรียญแต่ละแบบตามตัวอย่าง
```

ตัวอย่างการใช้งาน piggybank

```
p1 = piggybank()
print( int(p1) )        # 0

p1.add1(10)             # เพิ่มเหรียญ 1 บาท 10 เหรียญ
print( int(p1) )        # 10

p1.add2(5)              # เพิ่มเหรียญ 2 บาท 5 เหรียญ
print( int(p1) )        # 20

p1.add5(2)              # เพิ่มเหรียญ 5 บาท 2 เหรียญ
print( int(p1) )        # 30

p1.add10(1)             # เพิ่มเหรียญ 10 บาท 1 เหรียญ
print( int(p1) )        # 40

p2 = piggybank()
p2.add10(5)             # เพิ่มเหรียญ 10 บาท 5 เหรียญ

print( p1 < p2 )        # True


print( str(p1) )        # {1:10, 2:5, 5:2,
10:1}
print( p2 )             # {1:0, 2:0, 5:0, 10:5}
```

เมท็อด **__lt__** ถูกเรียกเมื่อเราใช้ตัวปฏิบัติการ **<** กับ **piggybank** สองตัว เพื่อเปรียบเทียบว่าตัวซ้ายน้อยกว่าตัวขวาหรือไม่

เมท็อด **__int__** ถูกเรียกเมื่อ **int(p)** ทำงาน โดยที่ **p** เป็น **piggybank** ได้ผลลัพธ์เป็น **int** แทนค่าของ **p**

เมท็อด **__str__** ถูกเรียกเมื่อ **str(p)** ทำงาน โดยที่ **p** เป็น **piggybank** ได้ผลลัพธ์เป็นสตริงแทนค่าของ **p**

การส่งตรวจ

ให้นำโปรแกรมข้างล่างนี้ ต่อท้าย **class piggybank** ที่เขียนข้างบนนี้ แล้วจึงส่งให้ Grader ตรวจ

```
cmd1 = input().split(';')
cmd2 = input().split(';')
p1 = piggybank(); p2 = piggybank()
for c in cmd1: eval(c)
for c in cmd2: eval(c)
```

ข้อมูลนำเข้า

คำสั่งต่าง ๆ เพื่อการทดสอบคลาส

ข้อมูลส่งออก

ผลการทำงานของโปรแกรมข้างบนที่อาศัยคลาส **piggybank**

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
`p1.add1(1);p1.add2(2);p1.add5(3);p1.add10(4)` `print(int(p1), str(p1))`	`60 {1:1, 2:2, 5:3, 10:4}`
`p1.add1(1);p1.add2(2);p1.add5(3);p1.add10(4)` `p2.add1(61); print(p1 < p2)`	`True`

12-07: กระปุมออมสิน 2

จงเขียนคลาส **piggybank** เพื่อผลิตอ็อบเจกต์กระปุกออมสิน ที่สามารถหยอดเหรียญ**มูลค่าอะไรก็ได้** โดยจำกัดจำนวนเหรียญ**รวมทุกแบบ**ในกระปุกแล้ว**ห้ามเกิน 100 เหรียญ** (ถ้าหยอดแล้วเกินไม่รับเพิ่ม) ตามโครงของคลาสและตัวอย่างการใช้งานข้างล่างนี้

โครงของคลาส piggybank

```
class piggybank:
  def __init__(self):
     # มีตัวแปร self.coins เก็บ dict  เริ่มต้นให้ว่าง ๆ
     # มี key เป็นมูลค่าเหรียญ และ value เป็นจำนวนเหรียญ

  def add(self, v, n):
     # ถ้าเพิ่มจำนวนเหรียญในกระปุกอีก n เหรียญแล้วเกิน 100
     #  จะไม่ให้เพิ่ม ให้คืน False แทนว่า เพิ่มไม่สำเร็จ
     # แปลง v เป็น float ก่อน (เพิ่ม 5 กับ 5.0 จะได้เหมือนกัน)
     # ถ้ากระปุกไม่เคยมีเหรียญ v ทำ self.coins[v]= 0
     # ทำคำสั่ง self.coins[v] += n
     # คืน True แทนว่าเพิ่มสำเร็จ

  def __float__(self):
     # นำค่าของเหรียญคูณกับจำนวนเหรียญ ของเหรียญทุกแบบ
     # ต้องคืนจำนวนแบบ float เท่านั้น อยากคืนศูนย์ ก็ต้อง 0.0

  def __str__(self):
     # คืนสตริงที่แสดงจำนวนเหรียญแต่ละแบบตามตัวอย่าง
     # โดยให้เรียงเหรียญตามมูลค่าจากน้อยไปมาก
```

ตัวอย่างการใช้งาน piggybank

```
p1 = piggybank()
print( int(p1) )          # 0

p1.add(0.25, 4)           # เพิ่มเหรียญ 25 สตางค์ 4 เหรียญ
print( float(p1) )        # 1.0

p1.add(0.50, 1)           # เพิ่มเหรียญ 50 สตางค์ 1 เหรียญ
print( float(p1) )        # 1.5

p1.add(10, 1)             # เพิ่มเหรียญ 10 บาท 1 เหรียญ
print( float(p1) )        # 11.5

print( p1 )               # {0.25:4, 0.5:1,
10.0:1}

print(p1.add(10, 1))      # True  เพิ่มได้
print( float(p1) )        # 21.5

print(p1.add(1,94))       # False เพิ่มไม่ได้ เกิน 100
เหรียญ
print( float(p1) )        # 21.5
```

เมท็อด **__float__** ถูกเรียกเมื่อ **float(p)** ทำงาน โดยที่ **p** เป็น **piggybank** ได้ผลลัพธ์เป็น **float** แทนค่าของ **p**

เมท็อด **__str__** ถูกเรียกเมื่อ **str(p)** ทำงาน โดยที่ **p** เป็น **piggybank** ได้ผลลัพธ์เป็นสตริงแทนค่าของ **p**

การส่งตรวจ

ให้นำโปรแกรมข้างล่างนี้ ต่อท้าย **class piggybank** ที่เขียนข้างบนนี้ แล้วจึงส่งให้ Grader ตรวจ

```
cmd1 = input().split(';')
cmd2 = input().split(';')
p1 = piggybank(); p2 = piggybank()
for c in cmd1: eval(c)
for c in cmd2: eval(c)
```

ข้อมูลนำเข้า

คำสั่งต่าง ๆ เพื่อการทดสอบคลาส

ข้อมูลส่งออก

ผลการทำงานของโปรแกรมข้างบนที่อาศัยคลาส **piggybank**

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
`p1.add(1.11,2); print(float(p1), p1)` `print(float(p2), p2)`	`2.22 {1.11:2}` `0.0 {}`
`p1.add(0.25,1);p1.add(5,1);p1.add(0.25,2);p1.add(5.0,1)` `print(float(p1), str(p1))`	`10.75 {0.25:3, 5.0:2}`
`p1.add(0.25,1); print(p1.add(0.25,100))` `print(p1.add(0.25,99)); print(float(p1))`	`False` `True` `25.0`

12-08: เลขโรมัน

ระบบจำนวนแบบเลขโรมันแทนด้วยตัวอักษร 7 ตัวคือ I, V, X, L, C, D, M ซึ่งแทนค่า 1, 5, 10, 50, 100, 500, 1000 ตามลำดับ (อ่านรายละเอียดของเลขโรมันในหน้าถัดไป) จงเขียนคลาส **roman** เพื่อใช้สร้างเลขโรมัน ดังแสดงในโครงคลาส และตัวอย่างการใช้งานข้างล่างนี้ (เพื่อความง่าย กำหนดให้เราสนใจเฉพาะเลขโรมันที่มีค่าเทียบเท่าค่า 1 ถึง 4999 เท่านั้น)

โครงของคลาส roman	ตัวอย่างการใช้งาน roman
`class roman :` `  def __init__(self, r):` `  def __lt__(self, rhs):` `  def __str__(self):` `  def __int__(self):` `  def __add__(self, rhs):`	`a = roman("MCCXXXIV")  # 1234` `b = rint("LXVI")  # 66` `print( a < b )  # False` `print( str(a) )  # MCCXXXIV` `print( int(a) )  # 1234` `c = a + b` `print( str(c) )  # MCCC`

เมท็อด **__lt__** ถูกเรียกเมื่อเราใช้ตัวปฏิบัติการ **<** กับ **roman** สองตัว เพื่อเปรียบเทียบว่าตัวซ้ายน้อยกว่าตัวขวาหรือไม่

เมท็อด **__str__** ถูกเรียกเมื่อคำสั่ง **str(a)** ทำงาน โดยที่ **a** เป็น **roman** ได้ผลลัพธ์เป็นสตริงที่แทนค่าของ **a**

เมท็อด **__int__** ถูกเรียกเมื่อคำสั่ง **int(a)** ทำงาน โดยที่ **a** เป็น **roman** ได้ผลลัพธ์เป็น **int** ที่แทนค่าของ **a**

เมท็อด **__add__** ถูกเรียกเมื่อเราใช้ตัวปฏิบัติการ **+** กับ **roman** สองตัว ได้ผลลัพธ์เป็น **roman** ใหม่ที่แทนผลบวกที่ได้

การส่งตรวจ

ให้นำโปรแกรมข้างล่างนี้ ต่อท้ายคลาส **roman** ที่เขียนข้างบนนี้ แล้วจึงส่งให้ Grader ตรวจ

```
t, r1, r2 = input().split()
a = roman(r1); b = roman(r2)
if   t == '1' : print(a < b)
elif t == '2' : print(int(a),int(b))
elif t == '3' : print(str(a),str(b))
elif t == '4' : print(int(a + b))
else          : print(str(a + b))
```

ข้อมูลนำเข้า

สตริง 3 ตัว คั่นด้วยช่องว่าง (ดูตัวอย่าง และโปรแกรมที่ส่งตรวจประกอบ)

ข้อมูลส่งออก

ผลการทำงานของโปรแกรมข้างบนที่อาศัยคลาส **roman** ที่เขียน

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
`1 III IV`	`True`
`1 IV III`	`False`
`2 MMMCMXCIX MMII`	`3999 2002`
`3 MCMLXXXVII MMCXXIV`	`MCMLXXXVII MMCXXIV`
`4 MM CMXCIX`	`2999`
`5 MMMMX CXXIX`	`MMMMCXXXIX`

Roman numerals

From Wikipedia, the free encyclopedia

"Latin numerals" redirects here. For counting in Latin, see Latin § Numbers.

The numeric system represented by **Roman numerals** originated in ancient Rome and remained the usual way of writing numbers throughout Europe well into the Late Middle Ages. Numbers in this system are represented by combinations of letters from the Latin alphabet. Roman numerals, as used today, are based on seven symbols:[1]

Symbol	I	V	X	L	C	D	M
Value	1	5	10	50	100	500	1,000

Roman numeric system

The numbers 1 to 10 are usually expressed in Roman numerals as follows:

I, II, III, IV, V, VI, VII, VIII, IX, X.

Numbers are formed by combining symbols and adding the values, so II is two (two ones) and XIII is thirteen (a ten and three ones). Because each numeral has a fixed value rather than representing multiples of ten, one hundred and so on, according to *position*, there is no need for "place keeping" zeros, as in numbers like 207 or 1066; those numbers are written as CCVII (two hundreds, a five and two ones) and MLXVI (a thousand, a fifty, a ten, a five and a one).

Symbols are placed from left to right in order of value, starting with the largest. However, in a few specific cases,[2] to avoid four characters being repeated in succession (such as IIII or XXXX), subtractive notation is used: as in this table:[3][4]

Number	4	9	40	90	400	900
Notation	IV	IX	XL	XC	CD	CM

- I placed before V or X indicates one less, so four is IV (one less than five) and nine is IX (one less than ten)
- X placed before L or C indicates ten less, so forty is XL (ten less than fifty) and ninety is XC (ten less than a hundred)
- C placed before D or M indicates a hundred less, so four hundred is CD (a hundred less than five hundred) and nine hundred is CM (a hundred less than a thousand)[5]

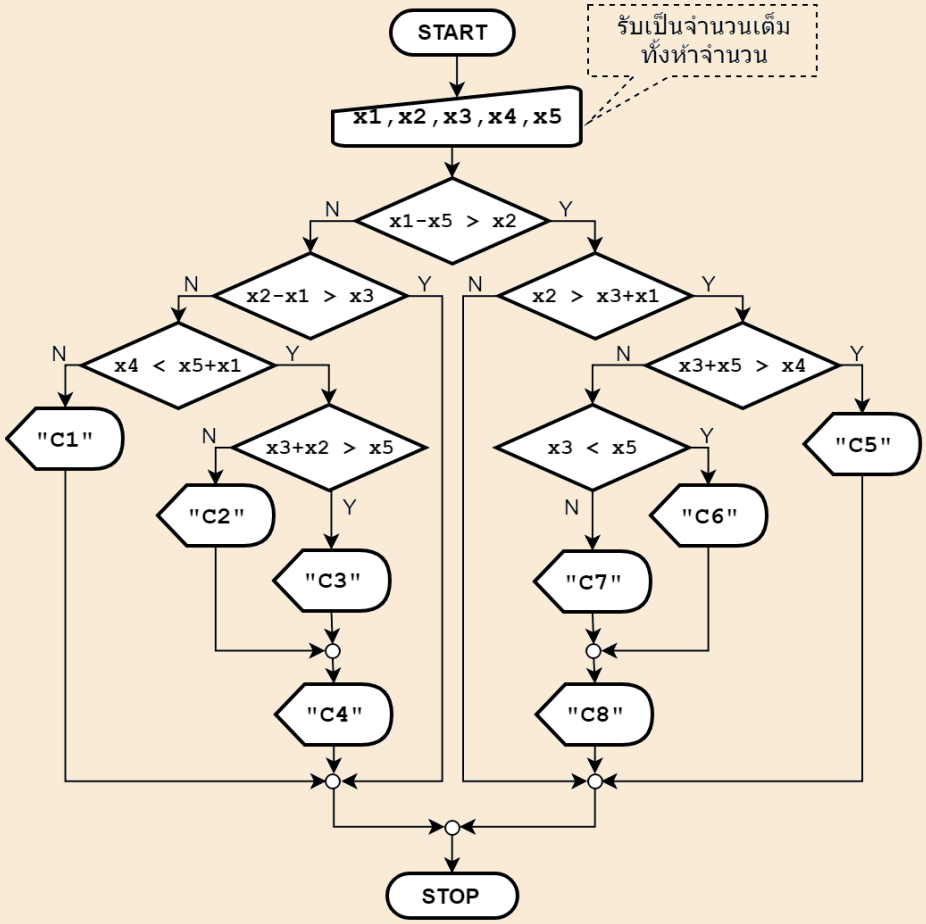
อ่านฉบับเต็มที่ **`https://en.wikipedia.org/wiki/Roman_numerals`**

แบบฝึกปฏิบัติเพิ่มเติม

https://pixabay.com/photos/cyber-glasses-virtual-virtual-world-1938449/

P-01: ผังงาน 1

จงเขียนโปรแกรมที่ทำงานตามผังงานข้างล่างนี้



ข้อมูลนำเข้า

หนึ่งบรรทัดประกอบด้วยจำนวนเต็มห้าจำนวนคั่นด้วยช่องว่าง

ใช้คำสั่ง `x1,x2,x3,x4,x5 = [int(e) for e in input().split()]`

ข้อมูลส่งออก

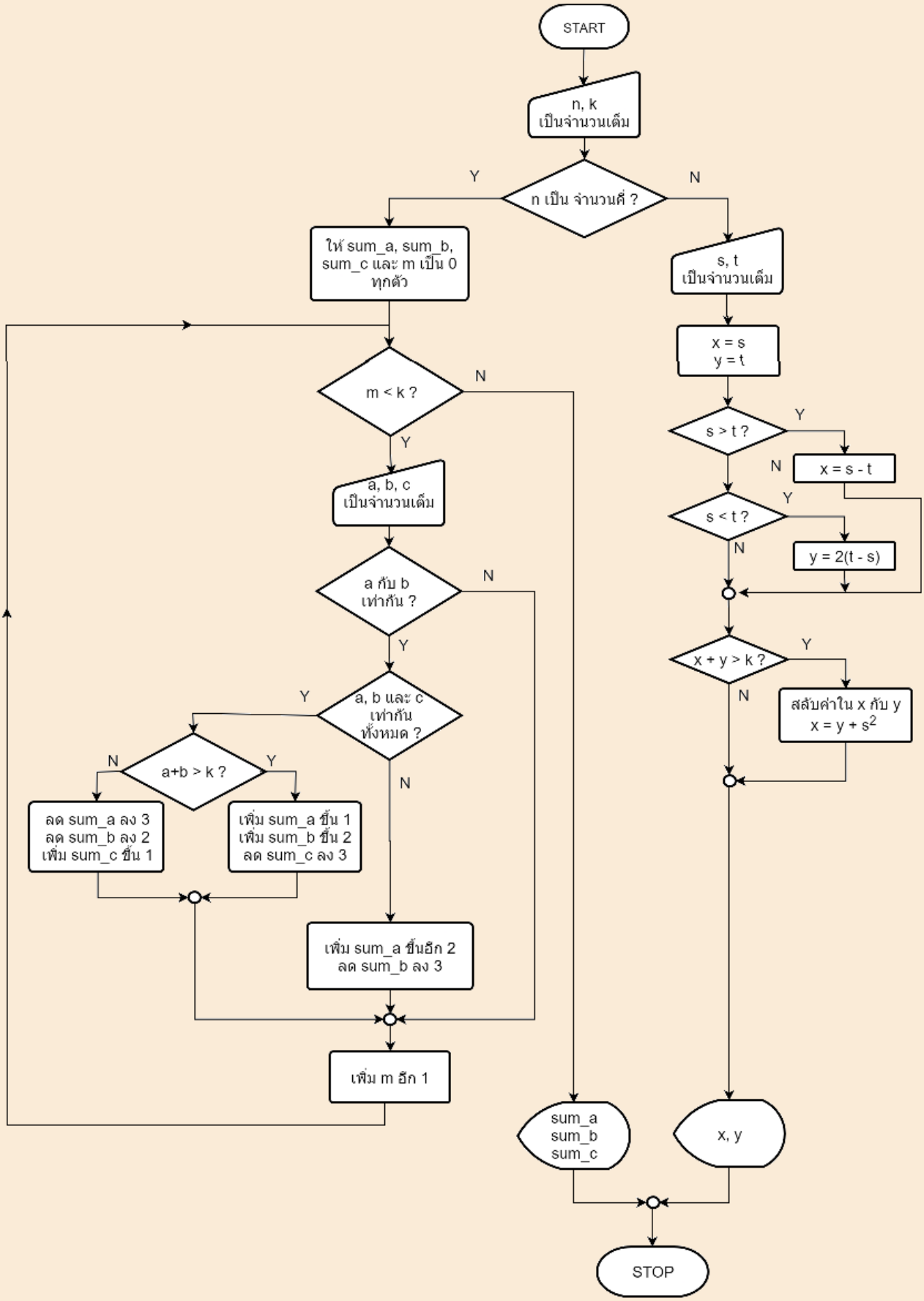
ตามที่แสดงในผังงาน

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
`4 1 3 5 2`	*ไม่มีผลลัพธ์*
`0 -15 -8 19 17`	`C1`
`-15 -8 19 0 17`	`C2` `C4`
`44 23 12 53 30`	`C3` `C4`
`20 1 -20 -10 18`	`C5`

P-02: ผังงาน 2

จงเขียนโปรแกรมที่ทำงานตามผังงานข้างล่างนี้



ข้อมูลนำเข้า

ตามที่แสดงในผังงาน

ข้อมูลส่งออก

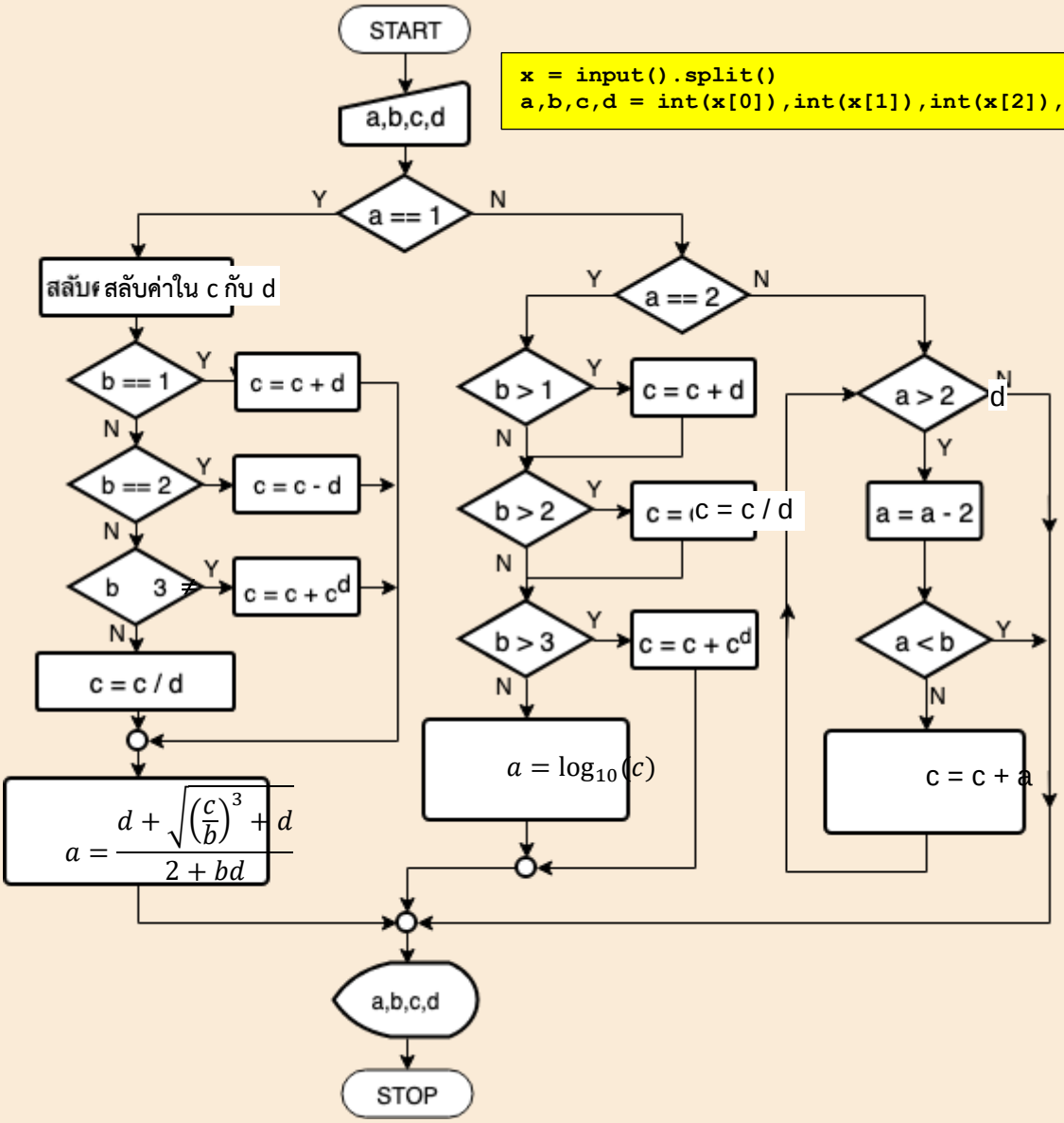
ตามที่แสดงในผังงาน

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
152 91 11 73	132 11
10 20 40 30	1610 10
431 3 10 10 10 1 12 1 1 2 1	1 2 -3
431 4 2 2 2 3 3 9 1 1 10 0 0 -9	3 -11 1

P-03: ผังงาน 3

จงเขียนโปรแกรมที่ทำงานตามผังงานข้างล่างนี้




```
x = input().split()
a,b,c,d = int(x[0]),int(x[1]),int(x[2]),int(x[3])
```

ข้อมูลนำเข้า

หนึ่งบรรทัดประกอบด้วยจำนวนเต็มสี่จำนวนคั่นด้วยช่องว่าง

ข้อมูลส่งออก

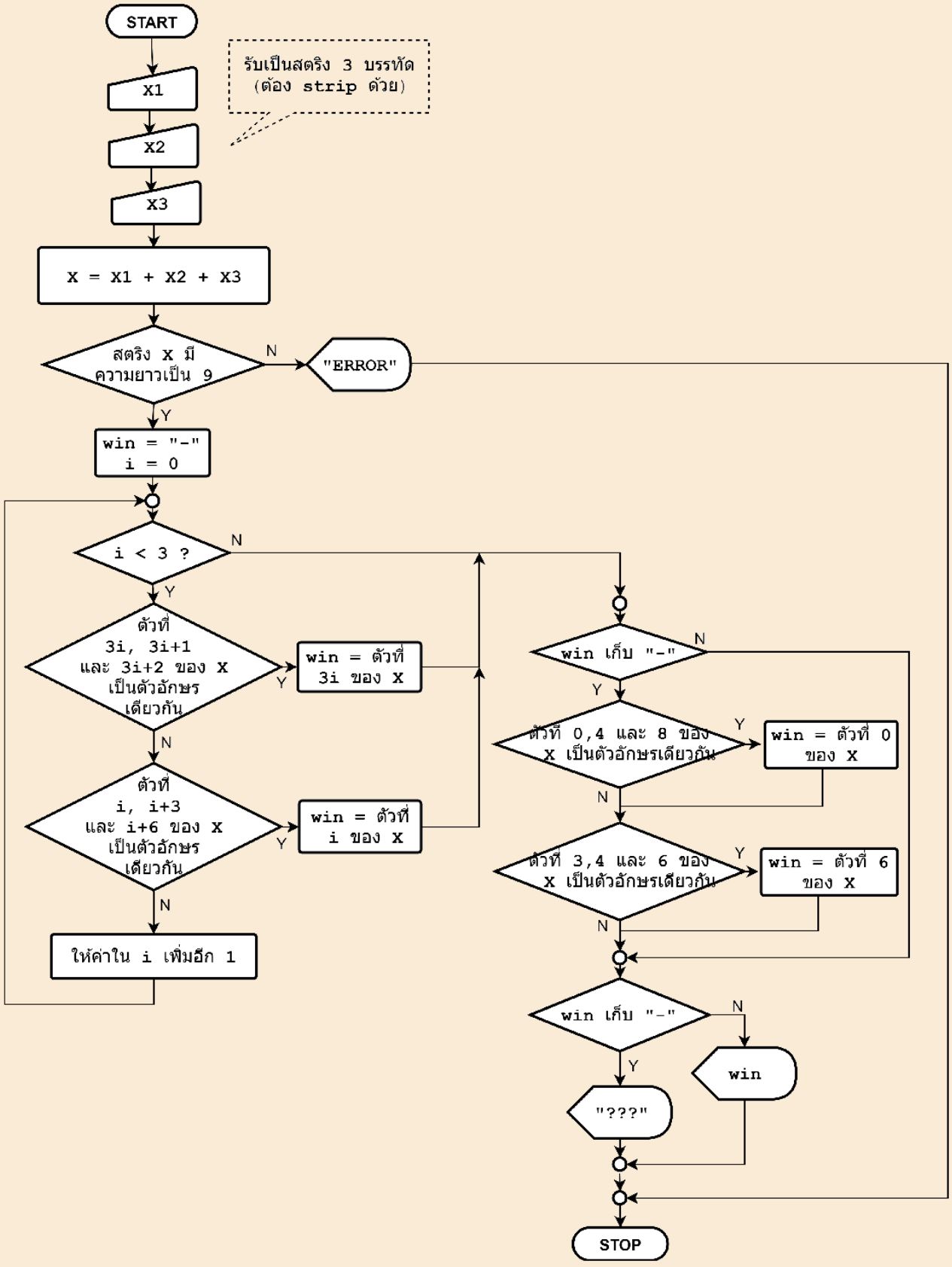
ตามที่แสดงในผังงาน

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
`1 2 7 5`	`0.5905931089239487 2 -2 7`
`2 2 2 2`	`0.6020599913279623 2 4 2`
`2 3 2 2`	`0.30102999566398114 3 2.0 2`
`100 20 9 1`	`18 20 2369 1`

P-04: ผังงาน 4

จงเขียนโปรแกรมที่ทำงานตามผังงานข้างล่างนี้



หมายเหตุ: อย่าลืมใช้ `input().strip()` เพื่อรับสตริงจากแป้นพิมพ์

ข้อมูลนำเข้า

สามบรรทัด ทุกบรรทัดเป็นสตริง

ข้อมูลส่งออก

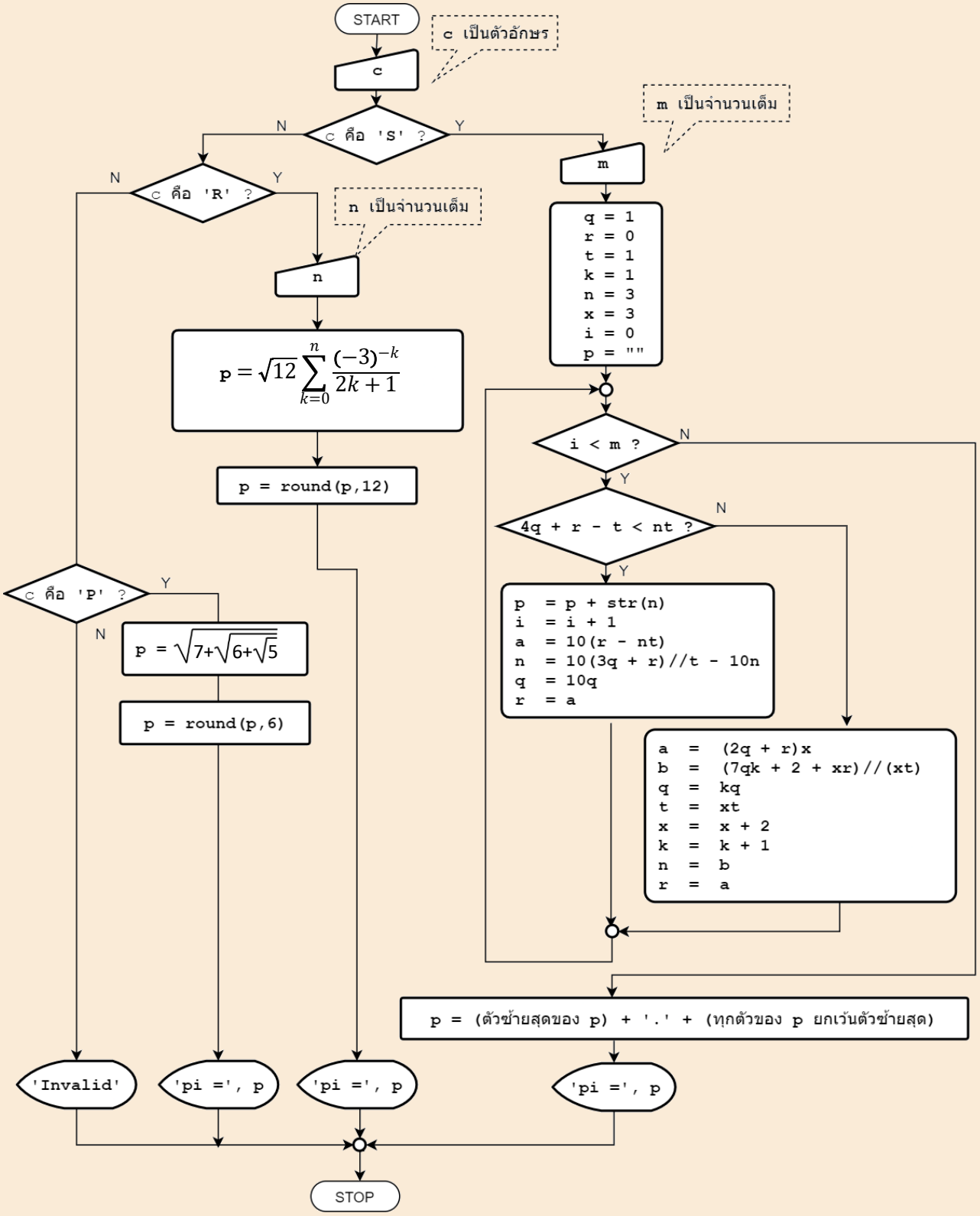
ตัวอักษร 1 ตัว, คำว่า **ERROR**, หรือไม่ก็สัญลักษณ์ **???**

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
\ / # o	ERROR
-OOO OXX XX	X
--X --- O--	???
OX- XOX --O	O
Ox--xO -x-	x
### --- $$$	#

P-05: ผังงาน 5

จงเขียนโปรแกรมที่ทำงานตามผังงานข้างล่างนี้



หมายเหตุ: อย่าลืมใช้ **`input().strip()`** เพื่อรับสตริงจากแป้นพิมพ์

`round(p,k)` จะคืนค่า **`p`** ปัดเลขหลังจุดทศนิยมให้มีเลขหลังจุดทศนิยม **`k`** ตำแหน่ง เช่น **`round(10/6,2)`** จะได้ **`1.67`** ให้สังเกตว่า ตัวแปร **`p`** ในสองกรณีทางซ้ายเป็นจำนวนจริง แต่ **`p`** ในกรณีทางขวาเป็นสตริง

ข้อมูลนำเข้า

บรรทัดแรก ตัวอักษร (ตัวที่สนใจคือ **R**, **S**, และ **P** ถ้าเป็นตัวอื่น **`Invalid`**)

บรรทัดที่สอง จำนวนเต็ม (เฉพาะกรณีที่ตัวอักษรในบรรทัดแรกคือ **R** หรือ **S**)

ข้อมูลส่งออก

ค่าประมาณของ π

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
`X`	**`Invalid`**
`R` **`10`**	**`pi = 3.141593304503`**
`S` **`3`**	**`pi = 3.14`**
`S` **`5`**	**`pi = 3.1415`**

P-06: ใครเป็นพี่

จงเขียนโปรแกรมรับ ชื่อเล่น เดือน วัน ปี เกิด ของคนสองคน แล้วแสดงว่าใครอายุมากกว่า

ข้อมูลนำเข้า

ข้อมูลชื่อและวันเดือนปีเกิดของสองคน คนละบรรทัด ในรูปแบบ

ชื่อเล่น ชื่อเดือน เลขวัน, เลขปี พ.ศ.

ข้อมูลส่งออก

ชื่อเล่นของผู้ที่เกิดก่อน ในกรณีที่อายุเท่ากัน ให้แสดงทั้งสองชื่อ เรียงตามลำดับที่อ่านเข้ามา

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
`Jane March 23, 2543` `Kate June 9, 2544`	`Jane`
`Jib July 4, 2545` `Mobile July 9, 2545`	`Jib`
`Noey December 10, 2532` `Jam December 10, 2532`	`Noey Jam`
`Jam December 10, 2532` `Noey December 10, 2532`	`Jam Noey`

ข้อแนะนำ

ชื่อเดือนภาษาอังกฤษ (คัดลอกไปใช้ในโปรแกรมได้)

```
January
February
March
April
May
June
July
August
September
October
November
December
```

P-07: เป่ายิ้งฉุบ

เกมเป่ายิ้งฉุบให้ผู้เล่นสองคนทำลักษณะมือพร้อมกันแทนกรรไกร กระดาษ หรือค้อน โดยให้ กรรไกรชนะกระดาษ กระดาษชนะค้อน และค้อนชนะกรรไกร จงเขียนโปรแกรมที่รับจำนวนเต็ม **m** จากนั้นก็รับผลการแข่งขันเกมเป่ายิ้งฉุบ ระหว่างผู้เล่น 2 คน โดยผู้ชนะการแข่งขันคือผู้ชนะเป่ายิ้งฉุบเป็นจำนวน **m** ครั้งก่อน ถ้าแข่งขันกันไปเป็นจำนวน **3m** เกม ยังสรุปไม่ได้ว่าใครชนะ ก็ให้ถือว่า **เสมอ** กัน

ข้อมูลนำเข้า (ดูตัวอย่างประกอบ)

บรรทัดแรกคือ **m** เป็นจำนวนเต็มบวก

บรรทัดถัด ๆ ไป แต่ละบรรทัดประกอบด้วยตัวอักษร 2 ตัว คั่นด้วยช่องว่าง โดย **R** แทนค้อน, **S** แทนกรรไกร และ **P** แทนกระดาษ ตัวอักษรแทนการทำมือของผู้เล่นหมายเลข 1 และตัวอักษรแทนการทำมือของผู้เล่นหมายเลข 2

ข้อมูลส่งออก (ดูตัวอย่างประกอบ)

มี 2 บรรทัด

- บรรทัดแรกแสดงจำนวนครั้งที่ผู้เล่นหมายเลข 1 ชนะ ตามด้วยจำนวนครั้งที่ผู้เล่นหมายเลข 2 ชนะ คั่นด้วยช่องว่าง
- บรรทัดที่สองสรุปว่าใครชนะ หรือเสมอกัน
  ถ้าผู้เล่นหมายเลข 1 ชนะ แสดง **`Player 1 wins`** ถ้าผู้เล่นหมายเลข 2 ชนะ แสดง **`Player 2 wins`**
  ถ้ายังไม่มีใครชนะได้ **m** ครั้ง หลังจากแข่งกันไป **3×m** เกม ก็ให้แสดง **`Tie`**

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
`1` `R P`	`0 1` `Player 2 wins`
`1` `R R` `P P` `S P`	`1 0` `Player 1 wins`
`1` `R R` `P P` `P P`	`0 0` `Tie`
`3` `R R` `P P` `S S` `P S` `P S` `P R` `S R`	`1 3` `Player 2 wins`
`3` `R R` `P P` `S S` `P S` `P S` `P P` `S S` `P P` `R S`	`1 2` `Tie`

P-08: โบว์ลิ่ง

โบว์ลิ่งเป็นกีฬาที่เราต้องโยนลูกโบว์ลิ่งลูกกลม ๆ ไปชนให้พินโบว์ลิ่งที่ตั้งไว้ 10 อันล้มให้มากสุด

เกมหนึ่งแบ่งออกเป็น 10 เฟรม (frame) แต่ละเฟรมโยนได้ 1 หรือ 2 ครั้ง (ยกเว้นเฟรม 10 อาจได้ถึง 3 ครั้ง)
การเขียนผลที่ได้ในแต่ละเฟรมทำดังนี้

- ถ้าโยนครั้งแรกในเฟรม ชนพินโบว์ลิ่งล้มหมด (เรียกว่า strike) เขียน **X** ในผลแล้วไม่ต้องโยนครั้งที่สอง
- ถ้าโยนครั้งแรกในเฟรม ชนพินโบว์ลิ่งล้มไม่หมด เขียนคะแนนตามจำนวนพินที่ล้ม แล้วก็โยนครั้งที่สอง
  - ถ้าครั้งที่สอง โยนแล้วพินที่เหลือล้มหมด (เรียกว่า spare) ให้เขียน **/** แต่ถ้าล้มไม่หมด ก็เขียนจำนวนที่ล้มในการโยนครั้งที่สอง
- สำหรับเฟรมที่สิบอาจได้โยนแถมอีก 1 หรือ 2 ครั้ง ถ้าครั้งแรก strike ได้โยนอีกสองครั้ง แต่ถ้าครั้งแรกล้มไม่หมด แต่ครั้งที่สองได้ spare ก็ได้โยนครั้งที่สาม

6	7	8
X	X	3 5
X+X+3		
10+10+3		
23		

การคิดคะแนน :

- เฟรมที่ได้ **X** ก็ได้ 10 แต้ม และยังได้คะแนนของการโยน 2 ครั้งถัดไป (ถ้ามี) มารวมเพิ่มในเฟรมนี้
- เฟรมที่ได้ **/** ก็ได้ 10 แต้ม และยังได้คะแนนของการโยน 1 ครั้งถัดไป (ถ้ามี) มารวมเพิ่มในเฟรมนี้
- เฟรมที่ไม่ได้ **X** และ **/** ก็ได้คะแนนตามที่โยนได้ในเฟรมนั้น

9	10
9 /	2 / 6
9+/+2	
9+1+2	
12	

- เฟรมที่สิบเป็นเฟรมสุดท้าย คะแนนที่ได้ในเฟรมนี้คือผลรวมของคะแนนการโยนในเฟรม (เช่นเฟรม 10 ได้ **XX3** ก็ได้ 23 คะแนน)

frame	1	2	3	4	5	6	7	8	9	10	Total
result	X	2 /	8 /	4 3	X	X	X	3 5	9 /	2 / 6	
	X+2+/	2+/+8	8+/+4	4+3	X+X+X	X+X+3	X+3+5	3+5	9+/+2	2+/+6	
	10+2+8	2+8+8	8+2+4	4+3	10+10+10	10+10+3	10+3+5	3+5	9+1+2	2+8+6	
score	20	18	14	7	30	23	18	8	12	16	166

หมายเหตุ : ในเกมโบว์ลิ่งจริง คะแนนที่เขียนในเฟรมใดจะเขียนเป็นคะแนนรวมสะสมตั้งแต่เฟรมแรกจนถึงเฟรมนั้น
แต่สำหรับโจทย์นี้ เราจะเขียนคะแนนเฉพาะของเฟรมนั้น ตามวิธีที่แสดงในตารางข้างบนนี้
จงเขียนโปรแกรมรับผลการโยนโบว์ลิ่ง เพื่อคำนวณคะแนนที่ต้องการ

ข้อมูลนำเข้า

บรรทัดแรกเป็นสตริงที่แทนผลการโยนโบวลิ่งทั้ง 10 เฟรม (สตริงที่ให้มานี้เป็นการเขียนผลการโยนโบว์ลิ่งที่ครบถ้วนถูกต้องแน่ ๆ)
บรรทัดที่สองเป็นจำนวนเต็ม ถ้าเป็น 1, 2, ..., 10 ให้แสดงคะแนนที่ได้ในเฟรมนั้น ถ้าเป็นจำนวนอื่น ให้แสดงคะแนนรวมของทั้งเกม

ข้อมูลส่งออก

คะแนนของเฟรม หรือคะแนนรวม ตามลักษณะข้อมูลนำเข้าที่ได้รับ

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
X2/8/43XXX359/2/6 0	166
9/9/9/9/9/9/9/9/9/9/1 -4	182
X2/8/43XXX359/2/6 6	23
00000000000000000X11 10	12
000000000000000000XX1 10	21
000000000000000000002/2 10	12
0000000000000000027 10	9

P-09: กอล์ฟ

กอล์ฟ คือ กีฬาชนิดหนึ่งที่ผู้เล่นต้องตีลูกกอล์ฟให้ลงหลุม โดยใช้จำนวนการตี หรือสโตรค (stroke) ให้น้อยสุด แต่ละหลุมมีการกำหนด "**พาร์**" (par) ซึ่งเป็นจำนวนสโตรคที่ผู้เล่นควรตีให้ลงหลุม เช่น หลุมพาร์สี่ ผู้เล่นควรตีให้ลงหลุมได้โดยการตี 4 ครั้ง แต่ละหลุมอาจมีพาร์ไม่เท่ากัน (เช่น พาร์สาม พาร์สี่ เป็นต้น) ในเกมกอล์ฟจริงมี 18 หลุม **แต่ในโจทย์นี้ มีแค่ 9 หลุม**

เพื่อให้นักกอล์ฟที่มีฝีมือต่างกันแข่งขันกันได้ จึงมีการคิด**แต้มต่อ** (handicap) ให้นักกอล์ฟ การคิดแต้มต่อมีหลายวิธี วิธีหนึ่งที่มีการใช้กันอยู่บ่อยๆ เรียกว่า **Double Peoria System** โดยต้องกำหนดก่อนว่า จะใช้หลุมใดมาพิจารณาคำนวณแต้มต่อ

ดังนั้น Input ของการคำนวณแต้มต่อประกอบด้วย พาร์ จำนวนสโตรค และสถานะการถูกเลือกหรือไม่เลือกมาพิจารณาแต้มต่อ ของแต่ละหลุม การคำนวณ**แต้มต่อ** และ**คะแนนสุทธิ** ทำตามขั้นตอนดังนี้

ดูตัวอย่างทางขวาประกอบ (ในช่อง เลือก/ไม่เลือก มีค่าเป็น 0 หรือ 1 ถ้าเป็น 1 คือถูกเลือก เป็น 0 คือไม่ถูกเลือก)

1. คำนวณ ค่าสโตรคปรับปรุง ซึ่งคือค่าน้อยกว่าระหว่าง จำนวนสโตรค กับ ค่าพาร์บวกอีกสอง (จากตัวอย่างคือช่องขวาสุด)
2. หาผลรวมของค่าสโตรคปรับปรุงของทุกหลุมที่ได้รับเลือก (จากตัวอย่างคือ ผลรวมของช่องขวาสุด ได้ 28)
3. หาผลรวมของพาร์ของทุกหลุม (จากตัวอย่างคือผลรวมของช่องพาร์ ได้ 36)
4. นำค่าในข้อ 2 คูณด้วย 1.5 แล้วลบด้วยค่าในข้อ 3 (จากตัวอย่างได้ 1.5×28 – 36 = 42 – 36 = 6)
5. นำค่าในข้อ 4 คูณด้วย 0.8 (จากตัวอย่างได้ 0.8×6 = 4.8)
6. นำค่าในข้อ 5 มา "*ปัดเศษ*" จะได้ **แต้มต่อ** *การปัดเศษค่า x ในที่นี้คือ*
   การหาจำนวนเต็มมากสุดที่มีค่าไม่มากกว่า x เช่น
   - *4.8 ปัดเศษแล้วได้ 4*
   - *–4.8 ปัดเศษแล้วได้ –5*
7. **คะแนนสุทธิ** เท่ากับ จำนวนสโตรครวม ลบด้วย แต้มต่อ

ตัวอย่าง

	input			
หลุม	พาร์	สโตรค	เลือก/ไม่เลือก	สโตรคปรับปรุง
1	4	4	1	4
2	4	5	1	5
3	5	4	1	4
4	3	4	0	
5	4	5	1	5
6	4	5	1	5
7	4	4	0	
8	3	6	1	5
9	5	6	0	
Σ	36	43		28

แต้มต่อ = *ปัดเศษ*(0.8×(1.5×28 – 36)) = 4

คะแนนสุทธิ = **จำนวนสโตรครวม – แต้มต่อ**

= 43 – 4 = 39

จงเขียนโปรแกรมรับข้อมูลของการเล่นกอล์ฟ 9 หลุม เพื่อคำนวณ จำนวนสโตรครวม แต้มต่อ และคะแนนสุทธิ

ข้อมูลนำเข้า

มีทั้งหมด 9 บรรทัดแทนข้อมูลของแต่ละหลุมจำนวน 9 หลุม แต่ละบรรทัดมี จำนวนเต็มบวก 3 จำนวน คั่นด้วยช่องว่าง จำนวนแรกคือ พาร์ของหลุมนี้ จำนวนที่สองคือ จำนวนสโตรคที่ใช้เพื่อตีลงหลุมนี้ จำนวนที่สุดท้ายเป็นค่า 1 หรือ 0 โดย 1 แทนการที่หลุมนี้ถูกเลือกในการคำนวณแต้มต่อ 0 แทนการไม่ถูกเลือก

ข้อมูลส่งออก

มีสามบรรทัด

- บรรทัดที่หนึ่งเป็นจำนวนสโตรครวม
- บรรทัดที่สองเป็นแต้มต่อ และ
- บรรทัดที่สามเป็นคะแนนสุทธิ ตามลักษณะข้อมูลนำเข้า

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
4 4 1 4 5 1 5 4 1 3 6 0 4 5 1 4 5 1 4 4 0 3 3 1 5 6 0	42 2 40
4 4 1 4 5 1 5 4 1 3 6 1 4 5 1 4 5 1 4 4 0 3 3 0 5 6 0	42 4 38
4 3 1 4 3 1 5 3 1 3 3 0 4 4 1 4 4 1 4 5 0 3 3 1 5 7 0	35 -5 40
4 3 1 4 3 1 5 3 0 3 3 0 4 4 1 4 4 1 4 5 0 3 3 1 5 7 1	35 0 35
4 4 1 4 4 1 5 5 0 3 3 0 4 4 1 4 4 1 4 4 0 3 3 1 5 5 1	36 0 36
4 5 1 4 5 1 5 6 0 3 4 0 4 5 1 4 5 1 4 5 0 3 4 1 5 6 1	45 7 38

P-10: การหมุนสตริง

จงเขียนโปรแกรมที่แสดงผลการหมุนชุดของสตริงโดยมีลักษณะการหมุน 3 แบบ ตามโอเปอร์เรเตอร์ที่กำหนด

ข้อมูลนำเข้า

บรรทัดแรกคือโอเปอร์เรเตอร์ โดยมี 3 โอเปอร์เรเตอร์ที่รองรับคือ 90, flip และ 180 แทนการหมุนชุดของสตริง 90 องศาตามเข็มนาฬิกา, การกลับด้านชุดของสตริงตามแนวซ้าย-ขวา และการหมุนชุดของสตริง 180 องศา

บรรทัดที่สองเป็นจำนวนบรรทัดของสตริงที่จะรับเข้า โดยสตริงแต่ละบรรทัดต้องมีจำนวนตัวอักษรเท่ากัน

บรรทัดที่ตามมาหลังจากนั้นจะเป็นข้อมูล 1 สตริงต่อบรรทัด

ข้อมูลส่งออก

ผลลัพธ์ของการหมุนชุดของสตริงตามโอเปอร์เรเตอร์ที่กำหนด (ดูตัวอย่างประกอบ)

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
90 3 ABCD 1234 WXYZ	W1A X2B Y3C Z4D
90 1 ABCD	A B C D
90 3 W X Y	YXW
90 3 AB ADEC PO2	Invalid size
flip 3 ABC DEF 123	CBA FED 321
180 2 ABCD 1234	4321 DCBA

ข้อแนะนำ

1) ลองเอาสตริงในแต่ละบรรทัดมาต่อกันให้หมดก่อน แล้วดูว่าถ้าต้องการหมุนข้อมูลตามตัวอย่าง ควรใช้ index เพื่อเข้าถึงค่าต่าง ๆ อย่างไร
2) ให้ตรวจสอบความผิดพลาดของ input กรณีเดียวคือ Invalid size คือสตริงที่อ่านเข้ามามีบางบรรทัดที่จำนวนตัวอักษรไม่เท่ากับบรรทัดอื่น

P-11: ฟังก์ชันอ๊อดอ๊อด

จงเขียนฟังก์ชันต่าง ๆ ข้างล่างนี้ ให้ทำงานตามหน้าที่ที่เขียนใน comment

```
def is_odd(n):
    # คืน (True/False) ว่า n เป็นจำนวนคี่หรือไม่


def has_odds(x):
    # คืน (True/False) ว่า x เป็นลิสต์ที่มีข้อมูลบางตัวเป็นจำนวนคี่


def all_odds(x):
    # คืน (True/False) ว่า x เป็นลิสต์ที่มีข้อมูลทุกตัวเป็นจำนวนคี่


def no_odds(x):
    # คืน (True/False) ว่า x เป็นลิสต์ที่มีไม่มีข้อมูลที่เป็นจำนวนคี่เลย


def get_odds(x):
    # คืนลิสต์ที่มีจำนวนคี่ที่มีเก็บในลิสต์ x (ลำดับก่อนหลังให้เป็นไปตามลำดับเดียวกับใน x)
    # เช่น x = [1,2,3,5,0] จะได้ผลคือ [1,3,5]


def zip_odds(a, b):
    # คืนลิสต์ที่สร้างจากการนำจำนวนคี่ใน a และ b มาสลับกันเก็บในลิสต์ผลลัพธ์ (เริ่มจากใน a ก่อน)
    # เช่น a = [0,8,1,2,4,6,5,7,9,2,3] กับ b = [4,19,11,12,10,17] จะได้คือ
    # [1,19,5,11,7,17,9,3]


exec(input().strip())
```

ข้อมูลนำเข้า

คำสั่งภาษา Python ที่ใช้ทดสอบการทำงานของฟังก์ชัน

ข้อมูลส่งออก

ผลที่ได้จากการสั่งทำงานคำสั่งที่ได้รับ

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
`print(is_odd(31))`	`True`
`print(has_odds([0,2,3,4,8]))`	`True`
`print(all_odds([1,3,11,17]))`	`True`
`print(no_odds([2,4,8]))`	`True`
`print(get_odds([1,3,11,2,17]))`	`[1, 3, 11, 17]`
`print(zip_odds([1,3,11,2,17], [2,4,97,99]))`	`[1, 97, 3, 99, 11, 17]`
`print(zip_odds([2,4,97,99], [1,3,11,2,17]))`	`[97, 1, 99, 3, 11, 17]`

P-12: ลำดับไพ่

ไพ่หนึ่งสำรับมีจำนวน 52 ใบ ประกอบด้วยไพ่ 4 ชุด ชุดละ 13 ใบ แต่ละชุดมีสัญลักษณ์ได้แก่ ดอกจิก (Clubs) ข้าวหลามตัด(Diamonds) โพแดง (Hearts) และ โพดำ (Spades) ในชุด 13 ใบ ประกอบด้วยตัวเลข 2 ถึง 10 และมี J (Jack) Q (Queen) K (King) และ A (Ace)

ขอแทนไพ่ 1 ใบด้วยตัวอักษร 2 ตัว ตัวแรกเป็นค่าไพ่ ตัวที่สองเป็นชุดไพ่

- **ค่าไพ่** เรียงตามลำดับจากค่าน้อยไปมากดังนี้ **A**, **2**, **3**, **4**, **5**, **6**, **7**, **8**, **9**, **T** (ใช้แทน **10**), **J**, **Q**,และ **K** นั่นคือ **A** มีค่าเป็น **1** และค่าของ **J**, **Q**, **K** เป็น **11**, **12**, และ **13** ตามลำดับ (ดูตารางข้างล่าง)
- **ชุดไพ่** เรียงตามลำดับจากค่าน้อยไปมากดังนี้ **C** (แทน Clubs), **D** (แทน Diamonds), **H** (แทน Hearts), และ **S** (แทน Spades) นั่นคือ **C**, **D**, **H**, และ **S** มีค่า **1**, **2**, **3**, และ **4** ตามลำดับ (ดูตารางข้างล่าง)

A	2	3	4	5	6	7	8	9	T	J	Q	K
1	2	3	4	5	6	7	8	9	10	11	12	13

C	D	H	S
1	2	3	4

ตัวอย่าง: **TS** หมายถึงไพ่ 10 Spades, **KD** หมายถึงไพ่ K Diamonds, **4H** หมายถึงไพ่ 4 Hearts, **4C** หมายถึงไพ่ 4 Clubs

สิ่งที่ต้องทำ

เขียนโปรแกรมรับลำดับของไพ่จำนวนหนึ่ง แล้วแสดงผลการเปรียบเทียบไพ่สองใบที่ติดกันในลำดับว่า ไพ่ใบซ้าย*ใหญ่กว่า*ใบขวา*เท่าใด* โดยมีวิธีเปรียบเทียบไพ่สองใบว่าใบหนึ่ง "ใหญ่กว่า" อีกใบ เท่าใด ดังนี้

1. เปรียบเทียบ**ค่าไพ่**ก่อน ไพ่ที่มีค่าไพ่มากกว่าจะถือว่าใหญ่กว่า และใหญ่กว่าตามผลต่างของค่าไพ่ (ค่าไพ่ใบซ้ายลบด้วยใบขวา)
2. ถ้าค่าไพ่เท่ากัน ให้ดูที่**ชุดไพ่** ไพ่ที่มีชุดไพ่มากกว่า ถือว่าใหญ่กว่า และใหญ่กว่าตามผลต่างของชุดไพ่ (ชุดไพ่ใบซ้ายลบด้วยใบขวา)

ตัวอย่าง: รับ input มาเป็น **ASKSKC5H5H** ซึ่งแทนลำดับไพ่ 5 ใบจากซ้ายไปขวา คือ **AS**, **KS**, **KC**, **5H** และ **5H** ได้ผลลัพธ์ดังนี้

ลำดับของไพ่ -->	AS	KS	KC	5H	5H	ผล	คำอธิบาย
คู่ที่ 1: เปรียบเทียบ	AS	KS				-12	**AS** เล็กกว่า **KS** อยู่ **12** (ดูค่าไพ่ **A** คือ **1**, **K** คือ **13**)
คู่ที่ 2: เปรียบเทียบ		KS	KC			+3	**KS** ใหญ่กว่า **KC** อยู่ **3** (ค่าไพ่เท่ากัน ดูชุดไพ่ **S** คือ **4**, **C** คือ **1**)
คู่ที่ 3: เปรียบเทียบ			KC	5H		+8	**KC** ใหญ่กว่า **5H** อยู่ **8** (ดูค่าไพ่ **K** คือ **13**)
คู่ที่ 4: เปรียบเทียบ				5H	5H	0	**5H** เท่ากับ **5H**
						ผลลัพธ์ที่แสดงคือ **-12+3+80**	

ข้อมูลนำเข้า

มี 1 บรรทัด เป็นสตริงที่แทนลำดับของไพ่อย่างน้อย 2 ใบ

ข้อมูลส่งออก

แสดงลำดับของผลลัพธ์การเปรียบเทียบไพ่ที่ติดกันใน input ว่า ไพ่ใบซ้ายใหญ่กว่าใบขวาเท่าใด ตามตัวอย่างข้างบน ในกรณีที่ไพ่ติดกันทั้งสองใบมีชุดและค่าเดียวกัน ให้แสดง **0**

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
AD2D3D4D	-1-1-1
5S5H5D5C5D5H5S	+1+1+1-1-1-1
ADADAD	00
ASKSKC5H5H	-12+3+80
AS2C4STCTDJS2S2S	-1-2-6-1-1+90

P-13: สักสี่ฟังก์ชัน

จงเขียนฟังก์ชันต่อไปนี้

convex_polygon_area(p)

- รับ: **p** เป็นลิสต์ที่เก็บจุด แต่ละจุดเป็นลิสต์สองช่องเก็บพิกัด x และ y ของจุดยอดต่าง ๆ กันของรูปหลายเหลี่ยมนูนออก
- คืน: พื้นที่ของรูปหลายเหลี่ยมนูนออก ดูวิธีการหาพื้นที่ของ convex polygon ได้ที่ https://www.mathwords.com/a/area_convex_polygon.htm

is_heterogram(s)

- รับ: **s** เป็นสตริง
- คืน: **True** ถ้า **s** เป็น heterogram ไม่เช่นนั้นก็คืน **False**

  หมายเหตุ: heterogram คือคำหรือวลีที่ถ้ามีตัวอักษรภาษาอังกฤษตัวใด ก็จะมีตัวนั้นแค่ตัวเดียว (ถือว่าตัวใหญ่กับตัวเล็กเหมือนกัน) เช่น **Python** เป็น แต่ **Java** ไม่เป็น (เพราะมี **a** สองตัว)

replace_ignorecase(s, a, b)

- รับ: **s**, **a** และ **b** เป็นสตริง
- คืน: สตริงที่ได้จากการแทนสตริงย่อยใน **s** ที่มีค่าเหมือนสตริงใน **a** (ไม่สนใจว่าเป็นตัวพิมพ์เล็กหรือใหญ่) ด้วยสตริง **b**

  เช่น **replace_ignorecase("Python is hard","Hard","easy")** ได้ **"Python is easy"**

  การหาค่าภายใน **s** ที่เหมือน **a** จะหาจากซ้ายไปขวา เช่น

  replace_ignorecase("AaAaA", "AA", "X") จะได้ **"XXA"**

top3(votes)

- รับ: **votes** เป็น dict ที่เก็บข้อมูลในรูปแบบ { *ชื่อดารา* : *คะแนนที่ได้รับในการโหวต* }
- คืน: ลิสต์ของสตริงที่เก็บชื่อดาราที่ได้รับคะแนนโหวตสูงสุด 3 อันดับแรก จากมากสุด (ซ้ายสุด) ไปหาน้อย ถ้าคะแนนเท่ากันให้เรียงตามชื่อจากน้อยไปมากตามพจนานุกรม (ถ้าข้อมูลที่ได้รับมีไม่ถึง 3 คน ก็ให้เก็บเท่าที่มี)

```
def convex_polygon_area(p):


def is_heterogram(s):


def replace_ignorecase(s, a, b):


def top3(votes):


# ต้องมีคำสั่งข้างล่างนี้ ตอนส่งให้ Grader ตรวจ
for k in range(2):
    exec(input().strip())
```

ทุกฟังก์ชันต้อง**ไม่เปลี่ยนแปลง**ค่าของพารามิเตอร์ที่รับมา

ข้อมูลนำเข้า

คำสั่งภาษา Python ที่ใช้ทดสอบการทำงานของฟังก์ชัน

ข้อมูลส่งออก

ผลที่ได้จากการสั่งทำงานคำสั่งที่ได้รับ

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
`print(convex_polygon_area([[0,0], [0,3], [4,0]]))` `print(convex_polygon_area([[0,0], [4,0], [0,3]]))`	`6.0` `6.0`
`print(is_heterogram("The big dwarf only jumps."))` `print(is_heterogram("Java"))`	`True` `False`
`print(replace_ignorecase("Python is hard", "Hard", "easy"))` `print(replace_ignorecase("AAabaAA", "Aa", "Aaa"))`	`Python is easy` `AaaabAaaA`
`v = {"A": 8888, "B": 6666, "C": 7777, "X":6666}` `print(top3(v))`	`['A', 'C', 'B']`

P-14: บริการส่งของ

ร้านขายของออนไลน์แห่งหนึ่งมีบริการจัดส่ง 4 รูปแบบคือ **Express**, **Quick**, **Normal** และ **Free** ซึ่งใช้เวลาในการจัดส่ง **1**, **3**, **7** และ **14** วันหลังจากสั่งซื้อตามลำดับ จงเขียนโปรแกรมรับข้อมูลการสั่งซื้อต่าง ๆ แสดงรายการของวันที่จะจัดส่งของถึงลูกค้า (อ่านรายละเอียดของผลลัพธ์ข้างล่าง)

หมายเหตุ: ถ้า เลขปี ค.ศ. หารด้วย 400 ลงตัว หรือ หารด้วย 4 ลงตัวแต่หารด้วย 100 ไม่ลงตัว เดือนกุมภาพันธ์ในปีนั้นก็มี 29 วัน เนื่องจากบริษัทนี้เพิ่งตั้งมาเมื่อปี พ.ศ. 2558 จึงถือว่าเลขปี พ.ศ. การสั่งซื้อต้องไม่น้อยกว่า 2558

การตรวจความถูกต้องของ input ให้**แจ้งแค่แบบเดียว** โดยตรวจตามลำดับ ดังนี้ (เจอผิดแบบบนก่อน ก็แจ้งผิดเลย ไม่ต้องตรวจต่อ ดูตัวอย่าง)

1. ถ้าเลขปีผิดเงื่อนไข ให้แสดง **`Invalid year`**
2. ถ้าเลขปีถูกต้อง แต่เลขเดือนผิด ให้แสดง **`Invalid month`**
3. ถ้าเลขปีกับเลขเดือนถูกต้อง แต่วันเดือนปีที่รับมาไม่มีในปฏิทิน เช่นวันที่ 31 เมษายน 2560 ให้แสดง **`Invalid date`**
4. ถ้าวันเดือนปีถูกต้อง แต่ประเภทการจัดส่งไม่ถูกต้อง (ตัวอักษรไม่ตรงกับที่กำหนดไว้ข้างต้น) ให้แสดง **`Invalid delivery type`**

ข้อมูลนำเข้า

หลายบรรทัด บรรทัดละหนึ่งคำสั่งซื้อ บรรทัดสุดท้ายจะเป็นคำว่า **`END`**

แต่ละบรรทัดประกอบด้วย เลขที่คำสั่งซื้อ ประเภทการจัดส่ง ตามด้วยจำนวนเต็ม 3 จำนวนแทนเลขวัน เลขเดือน และเลขปี พ.ศ. ที่สั่งซื้อ (ทุกส่วนคั่นด้วยช่องว่าง) ประเภทการจัดส่งคือ **`E`**, **`Q`**, **`N`** และ **`F`** แทนการจัดส่งแบบ **Express**, **Quick**, **Normal** และ **Free** ตามลำดับ

ข้อมูลส่งออก

มีจำนวนบรรทัดเท่ากับจำนวนคำสั่งซื้อที่รับมา แบ่งออกเป็นสองส่วน

- ส่วนแรกบอกว่า คำสั่งซื้อใดมีข้อผิดพลาดอะไร เรียงตามลำดับเดียวกับที่รับเข้ามา
- ส่วนที่สองบอกว่าคำสั่งซื้อที่ถูกต้องทั้งหลาย จะถูกจัดส่งในวันเดือนปีใด เรียงตามลำดับวันที่จะจัดส่งถึงผู้รับจากก่อนไปหลัง ถ้าส่งวันเดียวกัน ก็ให้เรียงตามเลขที่คำสั่งซื้อ

ดูตัวอย่าง สำหรับรายละเอียดและรูปแบบการแสดงผล (ขอให้สังเกต การเว้นวรรคในการแสดงผลด้วย)

ตัวอย่าง

input (จากแป้นพิมพ์)

```
10002 E 29 -200 2550
10003 X 30 -200 2559
10001 Q 31 4 2558
10004 N 29 5 2560
10005 Q 10 4 2559
10006 F 29 2 2560
10007 X 28 2 2560
10008 F 28 12 2560
10009 E 1 1 2559
10010 N 29 5 2560
END
```

output (ทางจอภาพ)

```
Error: 10002 E 29 -200 2550 --> Invalid year
Error: 10003 X 30 -200 2559 --> Invalid month
Error: 10001 Q 31 4 2558 --> Invalid date
Error: 10006 F 29 2 2560 --> Invalid date
Error: 10007 X 28 2 2560 --> Invalid delivery
type
10009: delivered on 2/1/2559
10005: delivered on 13/4/2559
10004: delivered on 5/6/2560
10010: delivered on 5/6/2560
10008: delivered on 11/1/2561
```

เรียงตามวันที่ถูกจัดส่ง ถ้าส่งวันเดียวกัน ก็เรียงตามเลขที่คำสั่งซื้อ

P-15: การลบจนได้วลีสลับอักษร

สตริง **A** เป็น anagram ของสตริง **B** ถ้าเราสามารถนำสลับลำดับตัวอักษรของ **A** แล้วให้เหมือนกับ **B** ได้ (ไม่สนใจว่าเป็นตัวเล็กตัวใหญ่ และไม่สนใจอักขระใด ๆ ที่ไม่ใช่ตัวอักษร) เช่น **"Debit card"** เป็น anagram ของ **"Bad credit !!!"** เป็นต้น

โจทย์ข้อนี้ต้องการทราบว่าจะต้องลบตัวอักษรอะไรบ้างในสตริง **A** กับ **B** เป็นจำนวนน้อยสุด เพื่อให้ **A** และ **B** เป็น anagram ของกันและกัน เช่น ให้ **A = "XXYZZZ"** และ **B = "ZYYX"** ต้องลบ **X** หนึ่งตัวและ **Z** สองตัวออกจาก **A** (เหลือ **"XYZ"**) และลบ **Y** หนึ่งตัวออกจาก **B** (เหลือ **"ZYX"**) ทำให้ที่เหลือเป็น anagram

ข้อมูลนำเข้า

สตริงสองบรรทัด

ข้อมูลส่งออก

ตัวอักษรที่ต้องลบออกจากสตริงของแต่ละบรรทัดที่ได้รับ (ลบเป็นจำนวนตัวน้อยสุด) เพื่อให้สตริงทั้งสองหลังการลบเป็น anagram (สนใจเฉพาะตัวอักษรเท่านั้น โดยให้ตัวเล็กเหมือนตัวใหญ่) ดูรูปแบบการแสดงผลลัพธ์จากตัวอย่างข้างล่างนี้

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
XxYZzZ **ZYYXyyy**	**XxYZzZ** **- remove 1 x** **- remove 2 z's** **ZYYXyyy** **- remove 4 y's** แสดง **'s** ด้วยในกรณีลบหลายตัว
-- Forty five -- **== Over fifty ==**	**-- Forty five --** **- None** **== Over fifty ==** **- None**
AAB **XYY**	**AAB** **- remove 2 a's** **- remove 1 b** **XYY** **- remove 1 x** **- remove 2 y's**
Gentleman **Elegant woman**	**Gentleman** **- None** **Elegant woman** **- remove 1 a** **- remove 1 o** **- remove 1 w** แสดงตัวอักษรที่ถูกลบเรียงตามลำดับในพจนานุกรม

P-16: รหัสมอส

รหัสมอสถูกใช้แทนข้อความด้วยจังหวะของเสียงหรือการกระพริบของแสง เช่นข้อความ **SOS** จะถูกแปลงเป็น `... --- ...` (จุดสามจุด ช่องว่าง ขีดสามขีด ช่องว่าง และจุดอีกสามจุด) ขอไม่อธิบายว่าจุด ขีด และช่องว่างแทนด้วยการส่งเสียงหรือแสงอย่างไร โจทย์นี้สนใจการแปลงข้อความเป็นสตริงของจุด ขีด กับช่องว่าง โปรแกรมข้างล่างนี้แปลงข้อความที่รับมาเป็นรหัสมอส

```
pattern = input().strip()
text = input().strip()
morse = ''
for e in text :
    j = pattern.find('[' + e + ']')
    j += 3
    k = pattern.find('[',j)
    morse += pattern[j:k] + ' '
print(morse.strip())
```

บรรทัดแรกผู้ใช้ต้องป้อน*รูปแบบการแปลงรหัสมอส*ของแต่ละตัวอักษร ตามด้วยข้อความที่ต้องการแปลงในบรรทัดที่สอง ตัวอย่างเช่นให้ *รูปแบบการแปลงรหัสมอส* คือ `[A].-[B]-...[C]-.-.[E].[` ตัวสีแดงคือตัวอักษร ตัวพื้นเหลืองคือรหัสมอสของตัวอักษรแดงทางซ้าย ส่วนเครื่องหมาย `[` และ `]` มีไว้แยกความแตกต่างว่าส่วนใดเป็นตัวอักษรและส่วนใดเป็นรหัสมอส จากตัวอย่างจะได้ **A** แปลงเป็น `.-` **B** แปลงเป็น `-...` **C** แปลงเป็น `-.-.` และ **E** แปลงเป็น `.` หากข้อความเป็น **AABBE** รหัสมอสก็คือ `.- .- -... -... .` (ให้สังเกตว่ามีช่องว่างคั่นทุกรหัสของตัวอักษร)

จงปรับปรุงโปรแกรมข้างบนนี้ ให้

1. อ่านจากแฟ้มข้อมูลแทนการอ่านข้อมูลจากแป้นพิมพ์ โดยอ่านชื่อแฟ้มจากแป้นพิมพ์
2. แปลงได้ทั้งจากข้อความเป็นรหัสมอส และรหัสมอสเป็นข้อความ โดยดูจากข้อความของบรรทัดแรกในแฟ้ม
   - ถ้าเป็น **T2M** คือให้โปรแกรมทำหน้าที่แปลงข้อความเป็นรหัสมอส ถ้าเป็น **M2T** คือให้โปรแกรมทำหน้าที่แปลงรหัสมอสเป็นข้อความ ถ้าไม่ใช่สองตัวนี้ ให้แสดง **Invalid code**
3. บรรทัดที่สองในแฟ้ม คือ สตริงแทน *รูปแบบการแปลงรหัสมอส*
4. บรรทัดที่เหลือในแฟ้ม คือข้อมูลที่ต้องการแปลง (ขึ้นกับประเภทของการแปลงที่ระบุในบรรทัดแรกว่าเป็น **T2M** หรือ **M2T**)
5. แสดงผลของการแปลงทางจอภาพ
6. ปรับให้โปรแกรมตรวจสอบด้วยว่า ถ้าเป็น **T2M** และพบตัวอักษรที่ไม่มีใน *รูปแบบการแปลงรหัสมอส* หรือ ถ้าเป็น **M2T** และพบรหัสที่ไม่มีใน *รูปแบบการแปลงรหัสมอส* ก็ให้แสดง **Invalid** ตามด้วยบรรทัดที่ต้องการแปลงแต่แปลงไม่ได้

ข้อมูลนำเข้า

ชื่อแฟ้ม (แฟ้มนี้มีอย่างน้อยสองบรรทัดแน่ ๆ)

ข้อมูลส่งออก

ผลการแปลงข้อมูลทางจอภาพ (ดูตัวอย่างประกอบ)

ตัวอย่าง

input (ทางแป้นพิมพ์)	แฟ้ม data.txt	output (ทางจอภาพ)
`data.txt`	`T2M` `[A].-[B]-...[C]-.-.[E].[` `A` `ABX` `BBE`	`.-` `Invalid : ABX` `-... -... .`
`data.txt`	`M2T` `[A].-[B]-...[C]-.-.[E].[` `.-` `.-. -.-.` `-... -... .`	`A` `Invalid : .-. -.-.` `BBE`
`data.txt`	`T2T` `[X]-..-[Y]-.--[Z]--..[9]---.[`	`Invalid code`

P-17: สนุ้กเกอร์

สนุกเกอร์ (snooker) คือเกมที่มีผู้เล่นสองคนพยายามแทงลูกสีขาวเพื่อไปชนลูกสีอื่นลงหลุมให้ได้คะแนนมากสุด ลูกสนุกเกอร์มีหลายสี แต่ละสีมีคะแนนกำกับดังนี้

สี	Red	Yellow	Green	Brown	Blue	Pink	Black
อักษรย่อแทนสี	R	Y	G	W	B	P	K
คะแนน	1	2	3	4	5	6	7

Six-red snooker (https://en.wikipedia.org/wiki/Six-red_snooker) เป็นสนุกเกอร์แบบหนึ่งที่มีลูกแดง 6 ลูก และสีอื่นอย่างละหนึ่งลูก กฎการแทงลูกลงหลุมมีเรื่องจุกจิกที่ขอไม่อธิบายในที่นี้ ขอสรุปการคิดคะแนนในการแข่งขันหนึ่งรอบหรือเรียกว่าหนึ่งเฟรม (frame) ดังนี้

- การแทงลูกมีสองช่วง
  - ช่วงแรก: ยังมีลูกแดงเหลือบนโต๊ะ ผู้เล่นแทงขาวชนแดงให้ลงหลุม ถ้าสำเร็จ จะได้คะแนนของลูกแดง จากนั้นจะเลือกลูกแถมสีอื่นสีใดก็ได้อีกหนึ่งลูก ที่จะแทงขาวให้ชนลูกที่เลือกให้ลงหลุม ถ้าสำเร็จ จะได้คะแนนของลูกสีนั้น และนำลูกสีนั้นกลับขึ้นมาวางบนโต๊ะ
  - ช่วงหลัง: ไม่มีลูกแดงบนโต๊ะ ผู้เล่นต้องแทงขาวชนลูกสีอื่น คือ เหลือง เขียว น้ำตาล น้ำเงิน ชมพู และดำ ตามลำดับ และได้คะแนนตามสีที่ลงหลุม
- ผู้เล่นคนใดแทงลูกไม่ได้ตามกฎ (ในบางกรณีอาจต้องเสียคะแนน) จะเปลี่ยนให้ผู้เล่นอีกคนเล่น (โจทย์นี้ ไม่สนใจการตัดคะแนน)

ตารางข้างล่างนี้แสดงตัวอย่างผลการแทงและคะแนนตามลำดับจากบนลงล่างในหนึ่งเฟรม ช่องขวาสุดแทนผลการแทง ที่จะใช้เป็น input ของโปรแกรมที่จะเขียนกัน เช่น **1RB** คือผู้เล่นหมายเลข 1 แทงขาวชนแดงลงหลุมสำเร็จ ตามด้วยลูกแถมสีฟ้าก็สำเร็จ ถ้าเป็น **X** แสดงว่าแทงไม่สำเร็จ

ผู้เล่นหมายเลข 1			ผู้เล่นหมายเลข 2			ข้อมูลขาเข้าของโปรแกรม
ลูกแรก	ลูกแถม	คะแนน	ลูกแรก	ลูกแถม	คะแนน	
Red	Blue	1+5 = 6				1RB
Red	Black	1+7 = 8				1RK
Red	ไม่สำเร็จ	1				1RX
			Red	Green	1+3 = 4	2RG
			Red	Pink	1+6 = 7	2RP
			Red	Brown	1+4 = 5	2RW
			ไม่สำเร็จ		0	2X
Red	Yellow	1+2 = 3				1RY
Yellow		2				1Y
Green		3				1G
Brown		4				1W
Blue		5				1B
ไม่สำเร็จ		0				1X
			Pink		6	2P
			Black		7	2K
รวม		32			29	

ข้อสังเกต: เกมจบเมื่อลูกแรกเป็นสีดำ

จงเขียนโปรแกรมรับผลการแทงตลอดทั้งเฟรม เพื่อคำนวณ คะแนนรวมของผู้เล่นทั้งสอง และสรุปผลการแข่งขัน

ข้อมูลนำเข้า

มีหลายบรรทัด จนกว่าจะจบเฟรม โดยแต่ละบรรทัดจะประกอบด้วยตัวอักขระ 2 หรือ 3 ตัว

- ตัวแรก แทนหมายเลขของผู้เล่น
- ตัวที่ 2 แทนสีของลูกที่ลงหลุม ถ้าเป็น **`X`** แสดงว่าแทงไม่สำเร็จ
- ตัวที่ 3 แทนสีของลูกแถมที่ลงหลุม ถ้าเป็น **`X`** แสดงว่าแทงไม่สำเร็จ (จะมีตัวที่ 3 ก็เมื่อตัวที่ 2 เป็น **`R`** คือลูกแดงลงหลุม ได้ลูกแถม)

ข้อมูลที่ระบบส่งมาให้ทดสอบมีตัวแรกเป็น **`1`** หรือ **`2`** เท่านั้น และตัวที่สองและสามเป็น **`R, Y, G, W, B, P, K`** หรือ **`X`** เท่านั้น

ข้อมูลส่งออก

มีสองบรรทัด

- บรรทัดแรก แสดงคะแนนรวมของผู้เลขหมายเลข 1 ตามด้วยคะแนนรวมของผู้เลขหมายเลข 2 (คั่นด้วยช่องว่าง 1 ช่อง)
- บรรทัดที่สอง แสดงข้อความ **`Player 1 wins`** หรือ **`Player 2 wins`** หรือ **`Tie`** เมื่อผู้เล่นหมายเลข 1 ชนะ หรือ ผู้เล่นหมายเลข 2 ชนะ หรือ เสมอกันตามลำดับ

ตัวอย่างาง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
`1RB` **`1RK`** **`1RX`** **`2RG`** **`2RP`** **`2RW`** **`2X`** **`1Y`** **`1G`** **`1W`** **`1B`** **`1X`** **`2P`** **`2K`**	**`29 29`** **`Tie`**
`1RB` **`1RP`** **`1RX`** **`2RB`** **`2RP`** **`2RW`** **`2X`** **`1Y`** **`1G`** **`1W`** **`1B`** **`1X`** **`2P`** **`2K`**	**`28 31`** **`Player 2 wins`**
`1X` **`2RK`** **`2RK`** **`2RK`** **`2RK`** **`2RK`** **`2RK`** **`2Y`** **`2G`** **`2W`** **`2B`** **`2P`** **`2K`**	**`0 75`** **`Player 2 wins`**

P-18: สอนฝรั่งอ่านเลขไทย

จงเขียนฟังก์ชันรับจำนวนเต็มที่มีค่าตั้งแต่ 0 ถึง 9999 เพื่อคืนสตริงคำอ่านไทย เช่น รับ **1024** ก็คืน **neung pun yi sip si**

หมายเหตุ: 101 อ่านว่า neung roi et (จำนวนที่ลงท้ายด้วย 1 ที่มีค่ามากกว่าหนึ่ง อ่านลงท้ายด้วย เอ็ด เสมอ)

```
def to_Thai( N ):
    # N เป็นจำนวนเต็มที่ค่ามีตั้งแต่ 0 ถึง 9999
    #   (ไม่ต้องตรวจว่าอยู่ในช่วงนี้หรือไม่ ข้อมูลทดสอบอยู่ในช่วงนี้แน่ ๆ)
    # คืนสตริงคำอ่านไทยของ N แต่ละคำคั่นด้วยช่องว่างหนึ่งช่อง (ดูตัวอย่างข้างล่าง)



exec(input().strip())
```

ใช้คำอ่านจากตารางข้างล่างนี้

soon	ศูนย์
neung	หนึ่ง
song	สอง
sam	สาม
si	สี่
ha	ห้า
hok	หก
chet	เจ็ด
paet	แปด
kao	เก้า
sip	สิบ
et	เอ็ด
yi	ยี่
roi	ร้อย
pun	พัน

ข้อมูลนำเข้า

คำสั่งภาษา Python ที่ใช้ทดสอบการทำงานของฟังก์ชัน

ข้อมูลส่งออก

ผลที่ได้จากการสั่งทำงานคำสั่งที่ได้รับ

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
`print(to_Thai(1))`	`neung`
`print(to_Thai(10))`	`sip`
`print(to_Thai(21))`	`yi sip et`
`print(to_Thai(101))`	`neung roi et`
`print(to_Thai(8009))`	`paet pun kao`

P-19: วิตามิน

จงเขียนโปรแกรมอ่านข้อมูลปริมาณวิตามินในผลไม้ต่าง ๆ เพื่อตอบคำถามตามคำสั่งต่าง ๆ ที่ได้รับ

ข้อมูลนำเข้า

- บรรทัดแรกคือค่า n ที่เป็นจำนวนเต็มบวกระบุจำนวนผลไม้
- n บรรทัดต่อมาเป็นข้อมูลของปริมาณวิตามินในผลไม้ n ชนิด บรรทัดละชนิด แต่ละบรรทัดประกอบด้วยชื่อผลไม้ ตามด้วยรายการของจำนวนจริงระบุปริมาณของวิตามินประเภทต่าง ๆ (ระบุประเภทวิตามินด้วยหมายเลข 1, 2, 3, ...)

  จำนวนวิตามินของผลไม้แต่ละชนิดมีเท่ากันแน่ ๆ
- บรรทัดสุดท้าย ประกอบด้วย สตริงคำสั่ง อาจตามด้วยจำนวนเต็มหรือสตริง ขึ้นกับ สตริงคำสั่งดังนี้
  - **show** แสดงข้อมูลผลไม้ทั้งหมด ตามลำดับที่อ่านเข้ามา
  - **get** *ชื่อผลไม้* แสดงรายละเอียดของ *ชื่อผลไม้*
    ถ้าไม่มี *ชื่อผลไม้* ในข้อมูลขาเข้า ให้แสดง *ชื่อผลไม้* not found (การเปรียบเทียบ *ชื่อผลไม้* ตัวพิมพ์ใหญ่เล็กถือว่าไม่เหมือนกัน)
  - **avg** m แสดงค่าเฉลี่ยของปริมาณวิตามินประเภท m ของผลไม้ทุกชนิด
  - **max** m แสดงชื่อผลไม้ที่มีปริมาณวิตามินประเภท m มากที่สุด (ถ้ามีมากสุดเกินหนึ่งชนิด ให้เลือกผลไม้ที่ชื่อปรากฏก่อนในพจนานุกรม) และแสดงปริมาณมากสุดนั้นด้วย
  - **sort** m แสดงชื่อผลไม้ตามลำดับปริมาณของวิตามินประเภท m จากน้อยไปมาก ถ้าปริมาณเท่ากันให้เรียงตามชื่อผลไม้จากน้อยไปมาก

ค่า m หรือ ประเภทวิตามินของข้อมูล จะเป็นจำนวนที่อยู่ในช่วงที่ถูกต้องแน่นอน โปรแกรมที่เขียนไม่ต้องตรวจสอบ

ข้อมูลส่งออก

ผลลัพธ์ที่แสดง ขึ้นกับสตริงคำสั่งที่ได้ ดูตัวอย่างประกอบ

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
3 apricots 0.2 0.06 apple 0.005 0.06 banana 0.008 0.04 show	apricots 0.2 0.06 apple 0.005 0.06 banana 0.008 0.04
3 apricots 0.2 0.06 0.05 0.05 apple 0.005 0.06 0.01 0.05 banana 0.008 0.04 0.03 0.01 get banana	banana 0.008 0.04 0.03 0.01
3 apricots 0.2 0.06 0.05 apple 0.005 0.06 0.01 banana 0.008 0.04 0.03 avg 1	0.071 ถ้าต้องการแสดงค่าของตัวแปร x ให้ปัดเศษหลังจุดทศนิยมก่อน แล้วค่อยแสดงผลด้วยคำสั่ง print(round(x,4))
3 apricots 0.2 0.06 0.05 0.05 apple 0.005 0.06 0.01 0.05 banana 0.008 0.04 0.03 0.01 max 2	apple 0.06
3 apricots 0.2 0.06 0.05 0.05 apple 0.005 0.06 0.01 0.05 banana 0.008 0.04 0.03 0.01 sort 4	banana apple apricots
3 apricots 0.2 0.06 apple 0.005 0.06 banana 0.008 0.04 get Durian	Durian not found

P-20: รหัส Gray

Gray codes คือลำดับของเลขที่เลขทุกคู่ที่ติดกันใด ๆ ในลำดับจะต่างกันแค่หลักเดียว เช่น ถ้าใช้ระบบเลขฐานสอง ลำดับ **00, 01, 11, 10** เป็น gray codes เพราะ **00** กับ **01** ต่างกันบิตเดียว **01** กับ **11** ต่างกันบิตเดียว และ **11** กับ **10** ก็ต่างกันบิตเดียว

เราสามารถสร้าง Gray codes ของเลขฐานสองขนาด 4 บิตได้ด้วยวิธีดังต่อไปนี้

- เริ่มที่ `0,1` เป็น Gray codes ขนาด 1 บิต

  เขียนกลับลำดับ ได้ `1,0`

  นำมาต่อกัน ได้ `0,1,1,0`

  เติม `0` ข้างหน้าของเลขทุกตัวในชุดซ้าย และเติม `1` ข้างหน้าของเลขทุกตัวในชุดขวา ได้

  `00,01,11,10` ได้ Gray codes ขนาด 2 บิต

- ทำต่อ จาก `00,01,11,10` เป็น Gray codes ขนาด 2 บิต

  เขียนกลับลำดับ ได้ `10,11,01,00`

  นำมาต่อกัน ได้ `00,01,11,10,10,11,01,00`

  เติม `0` ข้างหน้าของเลขทุกตัวในชุดซ้าย และเติม `1` ข้างหน้าของเลขทุกตัวในชุดขวา ได้

  `000,001,011,010,110,111,101,100` ได้ Gray codes ขนาด 3 บิต

- ทำต่อ จาก `000,001,011,010,110,111,101,100` เป็น Gray codes ขนาด 3 บิต

  เขียนกลับลำดับ ได้ `100,101,111,110,010,011,001,000`

  นำมาต่อกัน ได้ `000,001,011,010,110,111,101,100,100,101,111,110,010,011,001,000`

  เติม `0` ข้างหน้าของเลขทุกตัวในชุดซ้าย และเติม `1` ข้างหน้าของเลขทุกตัวในชุดขวา ได้

  `0000,0001,0011,0010,0110,0111,0101,0100,1100,1101,1111,1110,1010,1011,1001,1000`

  ได้ Gray codes ขนาด 4 บิต

จงเขียนโปรแกรมรับจำนวนเต็ม **n** กับ **k** เพื่อหา Gray codes ของเลขฐานสองขนาด **n** บิต และนำ codes ทั้งหมดที่หาได้มาแสดงบรรทัดละ **k** ตัว ในรูปแบบที่แสดงในตัวอย่างข้างล่างนี้ (ข้อสังเกต: Gray codes ของเลขฐานสองขนาด n บิต มีทั้งหมด 2^n ตัว)

ข้อมูลนำเข้า

บรรทัดแรกเป็นจำนวนเต็ม **n** (มีค่าไม่เกิน 15) และบรรทัดที่สองเป็นจำนวนเต็ม **k** (มีค่าไม่เกิน 100)

ข้อมูลส่งออก

บรรทัดแรกแสดงตำแหน่งของเลขฐานสองแต่ละตัว มีรูปแบบการแสดงตามตัวอย่างข้างล่างนี้

บรรทัดที่เหลือแสดง Gray codes ของเลขฐานสองขนาด **n** บิต บรรทัดละ **k** ตัว

ถ้า **n** ไม่ใช่จำนวนเต็มบวก ให้แสดง **Invalid n** ถ้า **k** น้อยกว่า 1 ให้แสดง **Invalid k** ถ้าผิดทั้งคู่ก็แสดง **Invalid n and k**

ตัวอย่าง

input	output (ทางจอภาพ)
-100 100	Invalid n
100 -100	Invalid k
-100 -100	Invalid n and k
2 8	1--2--3--4--5--6--7--8- 00,01,11,10
5 11	1-----2-----3-----4-----5-----6-----7-----8-----9-----10----11--- 00000,00001,00011,00010,00110,00111,00101,00100,01100,01101,01111 01110,01010,01011,01001,01000,11000,11001,11011,11010,11110,11111 11101,11100,10100,10101,10111,10110,10010,10011,10001,10000

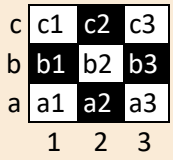
บรรทัดแรกแบ่งเป็น **k** ช่วง มีหมายเลขกำกับ

k-1 ช่วงแรกแสดง **n+1** ตำแหน่ง

ช่วงสุดท้ายแสดง **n** ตำแหน่ง

P-21: ตารางหมากฮอส

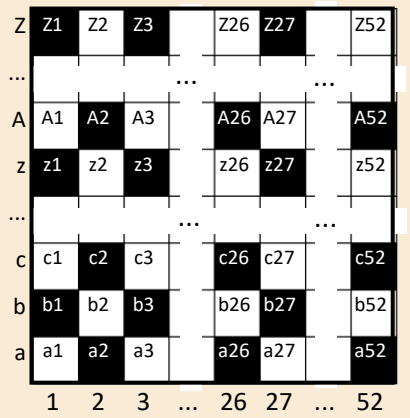
รูปข้างขวานี้คือตารางขนาด 3x3 ตำแหน่งของแต่ละช่องถูกกำหนดโดยใช้ตัวอักษร 1 ตัว (a, b และ c) แทนแถว ตามด้วยตัวเลขหนึ่งตัว (1, 2, และ 3) แทนคอลัมน์ และโปรแกรมข้างล่างนี้รับ ตำแหน่งช่อง (เช่น a3) เพื่อแสดงสีพื้นของช่องนั้น




```
position = input().strip()
row = position[0]
col = position[1]
r = 'abc'.find(row)
if r%2 == int(col)%2 :
    print('Black')
else:
    print('White')
```

หมายเหตุ: โปรแกรมนี้ ไม่ได้ตรวจสอบกรณีที่ป้อนตำแหน่งผิดไปจากที่ปรากฏในตาราง

สำหรับโปรแกรมที่ต้องเขียนในข้อนี้ ขอขยายตารางเป็น**ขนาด** 52x52 มี a ถึง z และ A ถึง Z เป็นตำแหน่งแถว และ 1 ถึง 52 เป็นตำแหน่งคอลัมน์ (ดูรูปทางขวานี้) จงเขียนโปรแกรมที่รับตำแหน่งช่อง เพื่อแสดงสีพื้นของช่องนั้น โดยต้องตรวจสอบความถูกต้องของตำแหน่งช่องด้วย



ข้อมูลนำเข้า

บรรทัดเดียวเป็นสตริง แทนตำแหน่งช่อง (มีขนาดอย่างน้อยหนึ่งตัวอักษร ตัวแรกไม่ใช่ blank แน่ๆ) ลักษณะข้อมูลขาเข้ามี 2 รูปแบบ

- แบบที่ 1: มีไม่เกิน 3 ตัว ตัวแรกแทนตัวอักษรแถว และตัวที่เหลือแทนเลขคอลัมน์ เช่น **A2**
- แบบที่ 2: มี 3 ตัวขึ้นไป จะมีรูปแบบ **row** = *ตัวอักษรแถว*, **col** = *เลขคอลัมน์* โดย
  - ส่วนที่แสดงด้วยพื้นเขียวมีใน input แน่นอน และมีปรากฏอย่างละหนึ่งครั้งแน่ ๆ (ยกเว้น = มีสองครั้ง) ตามลำดับที่แสดง (ยกเว้นคำว่า **row** กับ **col** อาจสลับตำแหน่งกันได้)
  - ช่องว่างคั่นมีได้ตั้งแต่ 0 ตัวขึ้นไป
  - ส่วน*ตัวอักษรแถว*และ*เลขคอลัมน์* อาจไม่มีหรือผิดได้

ข้างล่างนี้แทนตำแหน่ง **A2** ที่ถูกต้องทั้งสิ้น (สำหรับกรณี input ผิด ๆ ดูได้จากตัวอย่างข้างล่าง)

`col = 2 , row = A`	`col= 2, row =A`	`col =0002 , row = A`
`row= A, col = 02`	`col=2,row=A`	`col = 002 , row = A`

ข้อมูลส่งออก

- ถ้าตัวอักษรแถวไม่ถูกต้องหรือไม่มี (แต่เลขคอลัมน์ถูกต้อง) ให้แสดง **Invalid row** (ที่ถูกต้อง ต้องเป็น **a** ถึง **z**, **A** ถึง **Z**)
- ถ้าเลขคอลัมน์ไม่ถูกต้องหรือไม่มี (แต่ตัวอักษรแถวถูกต้อง) ให้แสดง **Invalid column** (ที่ถูกต้อง ต้องเป็น **1** ถึง **52**)
- ถ้าทั้งตัวอักษรแถวและเลขคอลัมน์ไม่ถูกต้องหรือไม่มี ให้แสดง **Invalid row and column**
- ถ้าตัวอักษรแถวและเลขคอลัมน์ถูกต้อง ให้แสดง **Black** หรือ **White** ให้ตรงกับสีพื้นของตำแหน่งนั้นในตาราง

ตัวอย่าง

input	output
`b2`	**White**
`z03`	**Black**
`col = 2 ,row = b`	**White**
`row=z, col = 03`	**Black**
`r3w`	**Invalid column**

AA	Invalid column
z76	Invalid column
9 9	Invalid row
$09	Invalid row
$ $	Invalid row and column
()	Invalid row and column
$59	Invalid row and column
%0X	Invalid row and column
row= r , col = 3 1	Invalid column
row = A, col = A	Invalid column
col = 76 , row=z	Invalid column
row = 9 , col = 9	Invalid row
row = AAA ,col=09	Invalid row
row = , col = $$	Invalid row and column
row =(, col =)	Invalid row and column
col = 59 , row=$	Invalid row and column
col= 1 1,row=%	Invalid row and column

P-22: การจัดรูปแบบการแสดงข้อความ

แฟ้มข้อมูลหนึ่ง ภายในเก็บข้อความ โดยช่องว่างทั้งหมดถูกแทนด้วยเครื่องหมายจุด และไม่มีจุดที่ต้นและปลายของทุกบรรทัดในแฟ้ม ดังตัวอย่างข้างล่างนี้

```
I.found.a.love.for.me..Darling.just.dive.right.in..And.follow.my.lead
Well.I.found.a.girl....beautiful.and.sweet..I....never.knew.you.were
the.someone.waiting.for.me
```

สิ่งที่ต้องการ โปรแกรมอ่านข้อความในแฟ้มมาแสดง โดย

- บรรทัดแรกแสดงตำแหน่งตัวอักษร (k ตำแหน่งตามที่ผู้ใช้ระบุ)
- แสดงได้บรรทัดละไม่เกิน k ตัวอักษร (แต่ก็มีข้อยกเว้นบ้าง)
- ต้องแสดงคำจากแฟ้มให้มากสุด ๆ ในแต่ละบรรทัด และ
- ไม่มีการแยกคำแสดงข้ามบรรทัด (แสดงทั้งคำในบรรทัดเดียวกัน)

เช่น ถ้า k = 40 จะได้ผลดังแสดงทางขวานี้ ให้สังเกตว่าจะไม่แสดงจุดที่ต้นและปลายบรรทัด ส่วนจุดภายในบรรทัดจะต้องเหมือนกับที่ปรากฏในแฟ้ม ให้สังเกตอีกนิดว่า การนำคำจากบรรทัดถัดไปในแฟ้มมาแสดงต่อ จะต้องคั่นด้วยจุด

```
---------1---------2---------3---------4
I.found.a.love.for.me..Darling.just.dive
right.in..And.follow.my.lead.Well.I
found.a.girl....beautiful.and.sweet..I
never.knew.you.were.the.someone.waiting
for.me
```

ในกรณีที่มีคำที่ยาวเกินค่า k ก็ต้องยอมให้แสดงคำนั้นได้ เช่น k = 6 เพื่อแสดงข้อความในแฟ้มข้างบนนี้ จะได้ผลทางขวานี้ (ขอแสดงไม่ครบทุกบรรทัด แค่ต้องการให้เห็นคำว่า **Darling** ที่ยาว 7 เกิน 6 ก็ต้องยอมให้เกิน)

```
------
I
found
a.love
for.me
Darling
(... ยังมีข้อความอีก ขอไม่แสดง ...)
```

ข้อมูลนำเข้า

บรรทัดแรกคือชื่อแฟ้ม บรรทัดที่สองเป็นจำนวนเต็มบวก **k** ตามที่อธิบายข้างต้น มีค่าน้อยกว่า 100 (แฟ้มข้อมูลที่กำหนดให้ อาจมีได้หลายบรรทัด แต่รับรองว่าไม่มีบรรทัดว่าง ๆ)

ข้อมูลส่งออก

ข้อความในแฟ้มที่แสดงทางจอภาพตามข้อกำหนดในโจทย์

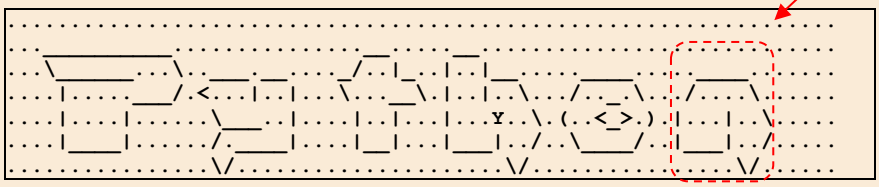
ตัวอย่าง

input (ทางแป้นพิมพ์)	แฟ้ม	output (ทางจอภาพ)
`data1.txt` `10`	`1.22...333.22.1.4444.22` `22.1.22.4444.22`	`---------1` `1.22...333` `22.1.4444` `22.22.1.22` `4444.22`
`data2.txt` `7`	`1.22.22.1.4444.22` `22.1.88888888`	`-------` `1.22.22` `1.4444` `22.22.1` `88888888`
`data1.txt` `26`	`1.22...333.22.1.4444.22` `22.1.22.4444.22`	`---------1---------2------` `1.22...333.22.1.4444.22.22` `1.22.4444.22`

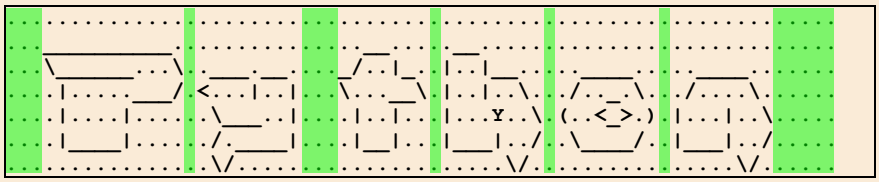
คำสั่ง **`s = s.strip(".")`** จะลบเครื่องหมายจุดที่อยู่ทางซ้ายและขวาของสตริง **s** ออกให้หมด เช่น ถ้า **`s = "...Ed.Sheeran.."`** คำสั่ง **`s = s.strip(".")`** จะทำให้ **s** เก็บสตริง **"Ed.Sheeran"**

P-23: ข้อความแอสกี

แฟ้มข้อมูลหนึ่งภายในเก็บข้อความหลาย ๆ บรรทัดที่ประกอบกันมองแล้วเป็นตัวอักษรใหญ่ ๆ (โดยช่องว่างทั้งหมดภายในแฟ้มจะถูกแทนที่ด้วยเครื่องหมายจุด) และ ทุกบรรทัดในแฟ้มมีจำนวนตัวอักขระเท่ากันหมด ดังตัวอย่างข้างล่างนี้ (เรียกตัวอักษรใหญ่ ที่สร้างจากการประกอบอักขระนี้ว่า ASCII Text)

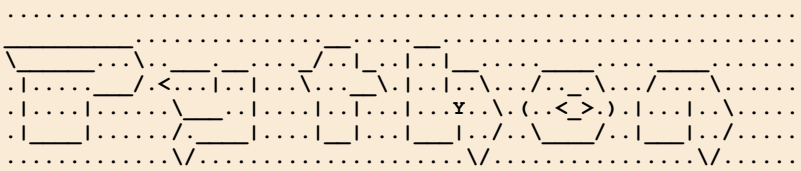


เราสามารถมองกลุ่มของเครื่องหมายจุดบางตัวข้างบนนี้ เสมือนเป็นช่องว่างที่คั่นระหว่างตัวอักษรใหญ่ได้ ดังแสดงด้วยพื้นสีเขียว (เรียกจุดเหล่านี้ว่า จุดที่แทนช่องว่าง)

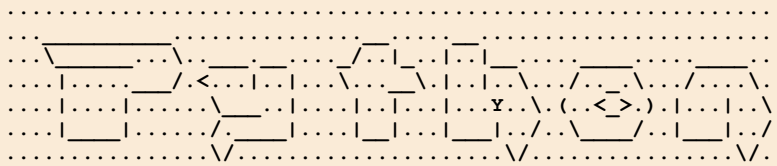


จงเขียนโปรแกรมอ่านแฟ้มที่ภายในเก็บข้อความที่มีตัวอักษรใหญ่มาแสดงทางจอภาพ โดยจะลบเครื่องหมายจุดที่แทนช่องว่างบางตัวออกก่อนแสดง ตามคำสั่งแสดงดังนี้

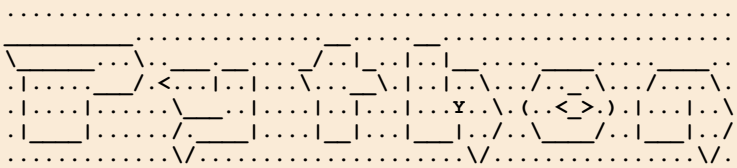
- ถ้าคำสั่งแสดงเป็น **LSTRIP** จะลบจุดที่แทนช่องว่างทางขอบซ้ายออกก่อนแสดง ได้ผลเป็น



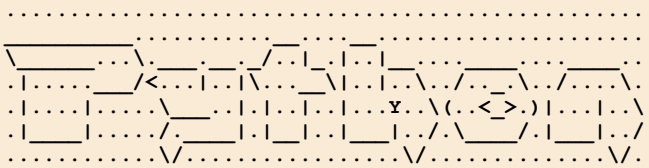
- ถ้าคำสั่งแสดงเป็น **RSTRIP** จะลบจุดที่แทนช่องว่างทางขอบขวาออกก่อนแสดง ได้ผลเป็น



- ถ้าคำสั่งแสดงเป็น **STRIP** จะลบจุดที่แทนช่องว่างทางขอบซ้ายและขอบขวาออกก่อนแสดง ได้ผลเป็น



- ถ้าคำสั่งแสดงเป็น **STRIP_ALL** จะลบจุดที่แทนช่องว่างทั้งหมด (ขอบซ้าย ขอบขวา และระหว่างตัวอักษรใหญ่) ออกก่อนแสดง ได้ผลเป็น



ข้อมูลนำเข้า

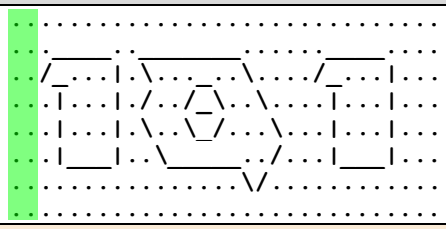
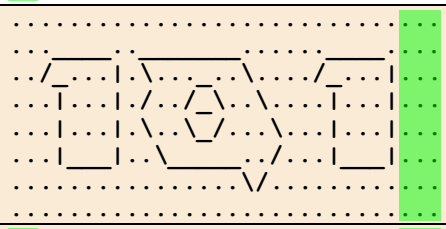
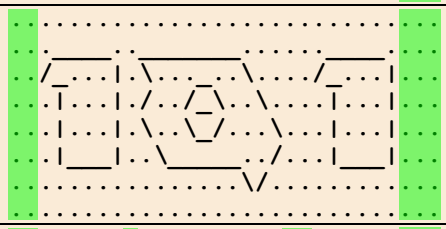
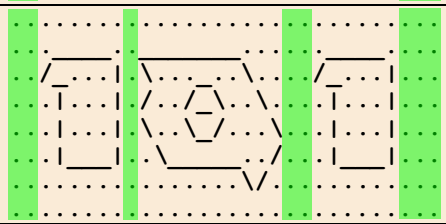
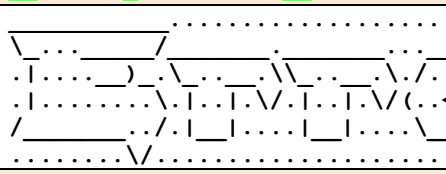
บรรทัดแรกคือชื่อแฟ้ม บรรทัดที่สองเป็นคำสั่งแสดง (แฟ้มที่ใช้ทดสอบ จะไม่มีกรณีที่เป็นจุดหมดทั้งแฟ้ม)

ข้อมูลส่งออก

ข้อความในแฟ้มที่แสดงทางจอภาพหลังลบจุดที่แทนช่องว่างตามคำสั่งแสดงที่กำหนด

ถ้าคำสั่งแสดงไม่ตรงกับที่กำหนด ให้แสดง **`Invalid command`**

ตัวอย่าง

input	แฟ้ม	output (ทางจอภาพ)
`101.txt` **`LSTRIP`**		
`101.txt` **`RSTRIP`**		
`101.txt` **`STRIP`**		
`101.txt` **`STRIP_ALL`**		
`error.txt` **`strip`**		**`Invalid command`**

P-22: ลำดับจำนวนในจัตุรัสเกลียว

จงเขียนฟังก์ชัน **spiral_square(n)** ซึ่งรับ **n** ที่เป็นจำนวนบวกคี่ แล้วคืนลิสต์ขนาด **n** ช่อง แต่ละช่องก็เป็นลิสต์ขนาด **n** ช่อง (มองได้ว่าเป็นตารางขนาด **n×n**) ภายในเก็บจำนวนเต็มมีค่า 1 ตรงกึ่งกลางตาราง แล้วเติมค่าเพิ่มขึ้นทีละหนึ่งในลักษณะวนเกลียวเป็นก้อนหอย ออกมารอบนอกจนครบทุกช่อง ดังตัวอย่างทางขวานี้เป็นตารางขนาด 7×7

```
37  36  35  34  33  32  31
38  17  16  15  14  13  30
39  18   5   4   3  12  29
40  19   6   1   2  11  28
41  20   7   8   9  10  27
42  21  22  23  24  25  26
43  44  45  46  47  48  49
```

```
def spiral_square(n):   # n is a positive odd number




def print_square(S):
    # เรียกใช้ฟังก์ชันนี้เพื่อแสดงค่าของ S ที่เป็นลิสต์ของลิสต์ของจำนวนเต็ม

    for i in range(len(S)):
        print(' '.join([(2*' '+str(e))[-3:] for e in S[i]]))

exec(input().strip())
```

ข้อมูลนำเข้า

คำสั่งภาษา Python ที่ใช้ทดสอบการทำงานของฟังก์ชัน

ข้อมูลส่งออก

ผลที่ได้จากการสั่งทำงานคำสั่งที่ได้รับ

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
`print_square(spiral_square(3))`	`  5  4  3` `  6  1  2` `  7  8  9`
`print_square(spiral_square(7))`	`37 36 35 34 33 32 31` `38 17 16 15 14 13 30` `39 18  5  4  3 12 29` `40 19  6  1  2 11 28` `41 20  7  8  9 10 27` `42 21 22 23 24 25 26` `43 44 45 46 47 48 49`

P-25: ดาราภาพยนตร์

มีข้อมูลขาเข้าเป็นชื่อภาพยนตร์และชื่อดารานำแสดงสองคน หลายเรื่อง จงเขียนโปรแกรมรับข้อมูลดังกล่าว และรับชื่อดารา เพื่อค้นว่าดาราเหล่านี้เล่นภาพยนตร์เรื่องใดบ้าง

ข้อมูลนำเข้า

บรรทัดแรกคือจำนวนภาพยนตร์ บรรทัดต่อๆ มามีจำนวนบรรทัดเท่ากับจำนวนภาพยนตร์ แต่ละบรรทัดประกอบด้วยชื่อเรื่อง และชื่อดารานำเสดงสองคน คั่นด้วยจุลภาค (comma) บรรทัดสุดท้ายคือรายการของชื่อดาราที่ต้องการค้น (คั่นด้วยจุลภาค)

ข้อมูลส่งออก

จากรายการชื่อดาราที่รับมาในบรรทัดสุดท้าย ให้แสดงผลลัพธ์เป็นบรรทัด ๆ แต่ละบรรทัดแสดงชื่อดารา ตามด้วยชื่อภาพยนตร์ทั้งหลายที่ดาราคนนี้แสดง (คั่นด้วย comma และ ช่องว่างหนึ่งช่อง ดูตัวอย่างประกอบ) ชื่อดาราใดที่หาภาพยนตร์ไม่พบเลย ให้แสดง **`Not found`**

ตัวอย่าง

input (จากแป้นพิมพ์)

```
9
Rush Hour, Jackie Chan, Chris Tucker
Shanghai Noon, Jackie Chan, Owen Wilson
Shanghai Knights, Jackie Chan, Owen Wilson
The Medallion, Jackie Chan, Lee Evans
Wedding Crashers, Owen Wilson, Vince Vaughn
Midnight in Paris, Owen Wilson, Rachel McAdams
Mousehunt, Nathan Lane, Lee Evans
The Forbidden Kingdom, Jackie Chan, Jet Li
The One, Jet Li, Jason Statham
Jet Li, Lee Evans, Tony Jaa
```

ให้แสดงชื่อภาพยนตร์เรียงตามที่ปรากฏในข้อมูลขาเข้า

output (ทางจอภาพ)

```
Jet Li -> The Forbidden Kingdom, The One
Lee Evans -> The Medallion, Mousehunt
Tony Jaa -> Not found
```

P-26: ระบบการค้นเอกสาร

ปัจจุบันระบบการค้นหาเอกสารหรือเวบไซต์ที่เกี่ยวข้องมีความสำคัญต่อสังคมยุคปัจจุบันเป็นอย่างมาก ระบบการค้นหาที่ดีนั้นนอกจากจะต้องมีความถูกต้องแล้ว ยังต้องสามารถเรียงลำดับการนำเสนอของเอกสารที่เกี่ยวข้อง เพื่อให้ผู้ใช้งานได้กดเลือกเอกสารที่เกี่ยวข้องได้สะดวกที่สุด เช่น เวบไซต์ google.com ที่พยายามจะหาเวบไซต์ที่ดีที่สุดมาสิบเวบไซต์เพื่อแสดงผลในหน้าแรก แม้ว่าจะมีเวบไซต์ที่เกี่ยวข้องเป็นหมื่นเป็นล้านเวบไซต์

โปรแกรมของโจทย์นี้ ค้นเอกสารที่มี**คะแนนความเกี่ยวข้องมากที่สุด** โดยคำนวณคะแนนความเกี่ยวข้องของเอกสารจากสูตร ดังนี้

$$\text{คะแนนความเกี่ยวข้อง} = \text{คะแนนความถี่ของคำ} \times \text{คะแนนความจำเพาะของคำ}$$

$$\text{คะแนนความถี่ของคำ} = \frac{\text{จำนวนครั้งที่คำที่ค้นหาปรากฏในเอกสาร}}{\text{จำนวนคำทั้งหมดในเอกสาร}}$$

$$\text{คะแนนความจำเพาะของคำ} = \frac{1}{\text{จำนวนคำที่ไม่ซ้ำกันทั้งหมดในเอกสาร}}$$

เช่น ถ้าเราต้องการค้นหาคำว่า **APPLE** และมีเอกสารทั้งหมดสองเอกสาร สามารถคำนวณคะแนนความเกี่ยวข้องได้ดังตารางนี้

ชื่อเอกสาร	ข้อความในเอกสาร	จำนวนคำ			คะแนนความเกี่ยวข้อง
		ทั้งหมด	ที่ไม่ซ้ำ	APPLE	
PPAP	I HAVE A PEN I HAVE AN **APPLE** AH PINEAPPLE PEN	11	8	1	$\frac{1}{11} \times \frac{1}{8}$ = **0.0113636**
MYAPPLE	**APPLE** WATCH MACBOOK AIR IPOD **APPLE** ORANGE	7	6	2	$\frac{2}{7} \times \frac{1}{6}$ = **0.047619**

พบว่า คะแนนความเกี่ยวข้องของ **MYAPPLE** มีค่าสูงสุด ผลลัพธ์ของการค้นคือเอกสาร **MYAPPLE**

หมายเหตุ เพื่อความง่าย ให้ถือว่า จะไม่มีกรณีที่คะแนนสูงสุดเท่ากัน

ข้อมูลนำเข้า

แบ่งเป็นสองส่วน

1. ส่วนเอกสาร: เริ่มด้วยตัวเลขจำนวนเอกสารที่จะใส่เข้าระบบหนึ่งบรรทัด ตามด้วยชื่อเอกสาร (หนึ่งบรรทัด) และข้อมูลในเอกสาร (หนึ่งบรรทัด) จนครบจำนวนเอกสารที่ระบุ
2. ส่วนค้นหา: ผู้ใช้จะใส่คำที่ต้องการค้นทีละบรรทัด ซึ่งโปรแกรมต้องแสดงผลการค้นหาทันที และรอรับคำค้นหาคำถัดไป หากผู้ใช้ใส่ **-1** ก็ให้จบการทำงาน

ให้ถือว่า ข้อมูลนำเข้าทั้งหมดเป็นตัวอักษรภาษาอังกฤษตัวพิมพ์ใหญ่ โดยข้อความในเอกสารคั่นแต่ละคำด้วยช่องว่างเท่านั้น

ข้อมูลส่งออก

ชื่อเอกสารที่มีคะแนนสูงสุด

หากคะแนนความเกี่ยวข้องที่สูงที่สุดคือ 0 ให้แสดง **NOT FOUND**

ตัวอย่าง

input (จากแป้นพิมพ์)

รับเข้าจนกว่าจะ

```
3
PPAP
I HAVE A PEN I HAVE AN APPLE AH PINEAPPLE PEN
MYAPPLE
APPLE WATCH MACBOOK AIR IPOD APPLE ORANGE
GOOGLE
GOOGLE MY WATCH AND GET MACBOOK AIR AND FREE I PHONE WATCH YOUTUBE
```

เหมือนตัวอย่าง

```
PEN
GOOGLE
APPLE
I
AKB
-1
```

Output (ทางจอภาพ)

```
PPAP
GOOGLE
MYAPPLE
PPAP
NOT FOUND
```

P-27: การประมูล

การประมูลเป็นการขายสินค้า โดยขายให้แก่ผู้ที่เสนอราคาสูงสุด โปรแกรมที่จะพัฒนานี้อนุญาตให้เปลี่ยนราคาประมูลได้ (ทั้งเพิ่มหรือลดราคาเสนอ) หรือจะถอนประมูลก็ได้

จงเขียนโปรแกรมที่รับรายการการเสนอราคาประมูล และการถอนการประมูลของผู้เข้าประมูลต่าง ๆ เพื่อสรุปว่า ใครประมูลได้สินค้าอะไร รวมเป็นเงินทั้งหมดเท่าไร ดังตัวอย่างข้างล่างนี้

Input	คำอธิบาย
8 B b3 p1 11 B b1 p2 10 B b2 p2 9 B b2 p4 1 W b1 p2 W b1 p4 B b1 p1 11 B b3 p4 5	บรรทัดแรกเป็น 8 ระบุว่ามีรายการที่เกี่ยวกับการประมูลตามมาอีก 8 รายการ 8 บรรทัดต่อมา แต่ละบรรทัดมีอักษรแรกเป็น B หรือ W ถ้าเป็นตัวอื่น ก็ข้ามบรรทัดนั้นไป ● ถ้าเป็น B คือการเสนอราคาประมูล (Bid) จะตามด้วย ***รหัสผู้ประมูล รหัสสินค้า*** และ***ราคาที่เสนอ*** การเสนอราคาประมูลในบรรทัดใด จะถือว่า เป็นการเสนอราคาประมูลล่าสุด คือ จะแทนราคาที่เคยเสนอไปก่อนนี้ (ถ้ามี) ● ถ้าเป็น W คือการถอนการประมูล (Withdraw) จะตามด้วย ***รหัสผู้ประมูล*** และ***รหัสสินค้า*** (ถ้าไม่พบรายการการประมูลที่มีรหัสผู้ประมูลและรหัสสินค้าที่จะถอน ก็ข้ามไป ไม่ต้องสนใจ) **ผู้ชนะประมูลคือ ผู้ที่เสนอราคาสูงสุด ในกรณีที่เสนอราคาเท่ากัน ให้ผู้ที่เสนอก่อนเป็นผู้ชนะ**

วิเคราะห์จากตัวอย่างข้างบนนี้ ได้การเปลี่ยนแปลงการประมูลต่าง ๆ แสดงดังตารางข้างล่างนี้

	สินค้าที่ถูกเสนอราคาประมูล		
input	p1	p2	p4
B b3 p1 11	(b3,11)		
B b1 p2 10	(b3,11)	(b1,10)	
B b2 p2 9	(b3,11)	(b1,10), (b2,9)	
B b2 p4 1	(b3,11)	(b1,10), (b2,9)	(b2,1)
W b1 p2	(b3,11)	(b2,9)	(b2,1)
W b1 p4	(b3,11)	(b2,9)	(b2,1)
B b1 p1 11	(b3,11), (b1,11)	(b2,9)	(b2,1)
B b3 p4 5	(b3,11), (b1,11)	(b2,9)	(b2,1), (b3,5)
ผู้ชนะประมูล	(b3,11) b1 และ b3 เสนอเท่ากัน แต่ b3 เสนอก่อน	(b2,9) b2 เสนอราคามากสุด	(b3,5) b3 เสนอราคามากสุด

สรุปว่า b1, b2 และ b3 ต้องชำระเงินรวมเป็น $0, $9 และ $16 ตามลำดับ

ดังนั้น โปรแกรมจะแสดงผลลัพธ์ของผู้ประมูล รายละบรรทัด เรียงตามรหัสผู้ประมูลจากน้อยไปมาก ในรูปแบบ

รหัสผู้ประมูล : $***จำนวนเงินรวมที่ต้องชำระ*** -> ***รายการสินค้าที่ประมูลได้*** (เรียงรหัสสินค้าจากน้อยไปมาก)

จากตัวอย่างข้างบนนี้ จะได้ผลที่แสดงคือ

```
b1: $0
b2: $9 -> p2
b3: $16 -> p1 p4
```

ข้อมูลนำเข้า

เป็นไปตามลักษณะที่อธิบายไว้ข้างต้น

ข้อมูลส่งออก

เป็นไปตามลักษณะที่อธิบายไว้ข้างต้น

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
แบบที่ 1 3 B b1 p4 4 B b2 p3 3 B b3 p2 2	b1: $4 -> p4 b2: $3 -> p3 b3: $2 -> p2
แบบที่ 2 3 B b1 p4 4 B b1 p3 3 B b2 p1 1	b1: $7 -> p3 p4 b2: $1 -> p1
แบบที่ 3 3 B b2 p1 1 B b1 p1 2 B b3 p1 3	b1: $0 b2: $0 b3: $3 -> p1
แบบที่ 4 3 B b3 p3 1 B b1 p3 2 B b1 p2 3	b1: $5 -> p2 p3 b3: $0

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
แบบที่ 5 4 B b1 p4 4 B b2 p3 3 W b1 p4 B b3 p2 2	b1: $0 b2: $3 -> p3 b3: $2 -> p2
แบบที่ 6 4 B b1 p4 4 B b1 p3 3 B b2 p1 1 W b2 p1	b1: $7 -> p3 p4 b2: $0
แบบที่ 7 4 B b2 p1 1 B b1 p1 2 B b3 p1 3 W b3 p1	b1: $2 -> p1 b2: $0 b3: $0
แบบที่ 8 5 B b3 p3 1 B b1 p3 2 B b1 p2 3 B b1 p1 1 W b1 p2	b1: $3 -> p1 p3 b3: $0
แบบที่ 9 8 B b1 p1 2 B b2 p1 2 B b3 p1 2 B b3 p2 99 B b2 p2 2 B b1 p2 2 W b3 p2 B b1 p4 4	b1: $6 -> p1 p4 b2: $2 -> p2 b3: $0

ข้อแนะนำ

ข้อกำหนดการประมูลมีหลายเงื่อนไข ข้อมูลขาเข้าที่ใช้ทดสอบจึงมีความง่ายยากหลายแบบ ดังนี้

รูปแบบข้อมูลที่ใช้ทดสอบการทำงานของโปรแกรม	การถอนประมูล	สินค้าแต่ละชิ้น มีผู้เสนอราคาประมูล N คน	ผู้ประมูลแต่ละคนเสนอราคาสินค้า M ชิ้น
แบบที่ 1	ไม่มี	$N \leq 1$	$M \leq 1$
แบบที่ 2	ไม่มี	$N \leq 1$	$M \geq 1$
แบบที่ 3	ไม่มี	$N \geq 1$	$M \leq 1$
แบบที่ 4	ไม่มี	$N \geq 1$	$M \geq 1$
แบบที่ 5	มี	$N \leq 1$	$M \leq 1$
แบบที่ 6	มี	$N \leq 1$	$M \geq 1$
แบบที่ 7	มี	$N \geq 1$	$M \leq 1$
แบบที่ 8	มี	$N \geq 1$	$M \geq 1$
แบบที่ 9	แบบที่ 8 + ผู้เสนอราคาสูงสุดมีหลายราย		

P-28: บีเอ็นเค

วง BNK มีจัดการเลือกตั้งเพื่อสรรหาสมาชิกที่จะได้ร้องเพลงในอัลบั้มใหม่ การโหวตในการเลือกตั้งของวง BNK นั้น โอตะ (เป็นคำเรียกแฟนคลับของวง BNK) สามารถโหวตได้ตามจำนวนซีดีที่ซื้อ นิสิตได้รับหน้าที่ให้จัดทำระบบวิเคราะห์การโหวตของเหล่าโอตะ

ในวง BNK นั้น จะมีไอดอลมากมายหลายคน โอตะคนหนึ่งอาจจะชื่นชอบไอดอลหลายคน การชื่นชอบไอดอลคนใดเป็นพิเศษ ก็จะเรียกว่า เป็นโอชิของไอดอลคนนั้น และสำหรับไอดอลที่โอตะคนหนึ่งชื่นชอบมากที่สุด ก็เรียกว่า เป็น**คามิโอชิ**ของโอตะคนนั้น

สำหรับโจทย์ข้อนี้ คามิโอชิจะดูจากคะแนนโหวต เช่น SOMCHAI โหวตให้ CHERPRANG กับ NATHERINE 10 กับ 5 คะแนน ตามลำดับ ดังนั้น SOMCHAI มีคามิโอชิคือ CHERPRANG ถ้าคะแนนโหวตเท่ากัน ถือว่า ไอดอลที่ชื่อมาก่อนตามพจนานุกรมเป็นคามิโอชิ

โจทย์ข้อนี้ต้องการให้นิสิตเขียนโปรแกรมที่แสดงชื่อไอดอล (สามลำดับแรกจากมากสุดไปน้อย) ตามหนึ่งในสามวิธีดังนี้

1. จัดอันดับไอดอลตามคะแนนโหวตรวมที่ได้รับ
2. จัดอันดับไอดอลตามจำนวนโอตะที่โหวตให้
3. จัดอันดับไอดอลตามจำนวนที่ได้เป็นคามิโอชิ

(กรณีทดสอบที่ใช้ จะได้ผลลัพธ์ในสามอันดับแรกที่มีคะแนนหรือจำนวนที่ใช้จัดอันดับที่ต่างกับอันดับอื่น นิสิตจึงไม่จำเป็นต้องเขียนโค้ดเพื่อรองรับกรณีเท่ากัน)

ข้อมูลนำเข้า

รับข้อมูลทีละบรรทัดประกอบด้วย ชื่อโอตะ ชื่อไอดอล คะแนนโหวต เช่น **`SOMCHAI CHERPRANG 10`**

บรรทัดสุดท้ายเป็นตัวเลข **1**, **2** หรือ **3** แทนผลลัพธ์ที่ต้องการให้แสดงไอดอลสามลำดับตามแบบที่กล่าวไว้ข้างต้น

- ทุก ๆ ชื่อของข้อมูลนำเข้าเป็นตัวพิมพ์ใหญ่ทั้งหมด
- ชื่อโอตะ สามารถซ้ำกันได้ และโอตะเดียวกันสามารถโหวตไอดอลคนเดิมได้มากกว่าหนึ่งครั้ง
- ข้อมูลที่ใช้ทดสอบจะมีไอดอลอย่างน้อยสามคนที่ได้รับการโหวตเสมอ

ข้อมูลส่งออก

ชื่อไอดอลสามคน คั่นด้วย **`,`** และเว้นวรรค เช่น **`CHERPRANG, NATHERINE, TURTLE`**

ตัวอย่างการโหวต	รวมผลคะแนนโหวต	คามิโอชิของโอตะแต่ละคน
SOMCHAI CHERPRANG 10 SOMCHAI NATHERINE 5 PRABHAS JENNIS 3 SOMCHAI CHERPRANG 4 DUANGDAO TURTLE 1 EKAPOL TURTLE 1 SETHA TURTLE 1 CHAIRAT TURTLE 1 JENNIS JENNIS 10 PRABHAS JANE 8 MANA CHERPRANG 2 MANEE CHERPRANG 1 CHUJAI TURTLE 1 PITI JENNIS 1 PITI JANE 1 VEERA CHERPRANG 1	CHERPRANG 18 JENNIS 14 JANE 9 TURTLE 5 NATHERINE 5 JANE ชื่อมาก่อน JENNIS แม้ว่าจำนวนโหวตเท่ากัน)	SOMCHAI: CHERPRANG PRABHAS: JANE DUANGDAO: TURTLE EKAPOL: TURTLE SETHA: TURTLE CHAIRAT: TURTLE JENNIS: JENNIS MANA: CHERPRANG MANEE: CHERPRANG CHUJAI: TURTLE PITI: JANE VEERA: CHERPRANG
	รวมผลจำนวนโอตะที่โหวตให้	รวมผลจำนวนที่ได้เป็นคามิโอชิ
	TURTLE 5 (SETHA, EKAPOL, DUANGDAO, CHAIRAT, CHUJAI) CHERPRANG 4 (MANEE, VEERA, SOMCHAI, MANA) JENNIS 3 (JENNIS, PITI, PRABHAS) JANE 2 (PRABHAS, PITI) NATHERINE 1 (SOMCHAI)	TURTLE 5 (SETHA, EKAPOL, DUANGDAO, CHAIRAT, CHUJAI) CHERPRANG 4 (MANEE, VEERA, SOMCHAI, MANA) JANE 2 (PRABHAS, PITI) JENNIS 1 (JENNIS) NATHERINE 0

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
SOMCHAI CHERPRANG 10 SOMCHAI NATHERINE 5 PRABHAS JENNIS 3 SOMCHAI CHERPRANG 4 DUANGDAO TURTLE 1 EKAPOL TURTLE 1 SETHA TURTLE 1 CHAIRAT TURTLE 1 JENNIS JENNIS 10 PRABHAS JANE 8 MANA CHERPRANG 2 MANEE CHERPRANG 1 CHUJAI TURTLE 1 PITI JENNIS 1 PITI JANE 1 VEERA CHERPRANG 1 1	CHERPRANG, JENNIS, JANE
SOMCHAI CHERPRANG 10 SOMCHAI NATHERINE 5 PRABHAS JENNIS 3 SOMCHAI CHERPRANG 4 DUANGDAO TURTLE 1 EKAPOL TURTLE 1 SETHA TURTLE 1 CHAIRAT TURTLE 1 JENNIS JENNIS 10 PRABHAS JANE 8 MANA CHERPRANG 2 MANEE CHERPRANG 1 CHUJAI TURTLE 1 PITI JENNIS 1 PITI JANE 1 VEERA CHERPRANG 1 2	TURTLE, CHERPRANG, JENNIS
SOMCHAI CHERPRANG 10 SOMCHAI NATHERINE 5 PRABHAS JENNIS 3 SOMCHAI CHERPRANG 4 DUANGDAO TURTLE 1 EKAPOL TURTLE 1 SETHA TURTLE 1 CHAIRAT TURTLE 1 JENNIS JENNIS 10 PRABHAS JANE 8 MANA CHERPRANG 2 MANEE CHERPRANG 1 CHUJAI TURTLE 1 PITI JENNIS 1 PITI JANE 1 VEERA CHERPRANG 1 3	TURTLE, CHERPRANG, JANE

P-29: ใช้ NumPy ไม่ใช้วงวน

จงเขียนฟังก์ชันต่าง ๆ ข้างล่างนี้ ให้ทำงานตามหน้าที่ที่เขียนใน comment โดยไม่ต้องใช้คำสั่งวงวนใด ๆ

```
import numpy as np

def eq(A, B, p):
    # A และ B เป็นอาเรย์ที่มีขนาดเท่ากัน (กี่มิติก็ได้), p เป็นจำนวนระหว่าง 0 ถึง 100
    # คืน True ถ้าข้อมูลใน A กับใน B ที่ตำแหน่งเดียวกัน
    #          มีค่าเท่ากันอย่างน้อยร้อยละ p ของจำนวนช่องทั้งหมด
    # ถ้ามีไม่ถึงก็คืน False


def closest_point_indexes(points, p):
    # points คืออาเรย์สองมิติที่มี 2 คอลัมน์ คอลัมน์ 0 เก็บพิกัด x คอลัมน์ 1 เก็บพิกัด y
    # p คืออาเรย์มิติเดียว 2 ช่อง เก็บพิกัด x และ y
    # คืน อาเรย์ที่เก็บ index ของจุดต่าง ๆ ใน points ที่อยู่ใกล้กับจุด p มากสุด
    #     ถ้ามีหลายจุดที่ใกล้สุดเท่ากัน ให้เก็บ index ทั้งหลายเรียงจากน้อยไปมาก


def number_of_inversions(A):
    # A เป็นอาเรย์หนึ่งมิติเก็บจำนวนเต็ม
    # คืน จำนวนคู่ข้อมูลใน A ที่ตัวทางซ้ายมากกว่าตัวขวา
    # (คือมีข้อมูล A[i] กับ A[j] ที่ i < j แต่ A[i] > A[j] อยู่กี่คู่)
    # เช่น [1 2 9 4 8 7] มี 4 คู่คือ 9 กับ 4, 9 กับ 8, 9 กับ 7 และ 8 กับ 7
    #     [9 7 5 3 2] มี 10 คู่ เพราะทุกคู่มีตัวซ้ายมากกว่าตัวขวา ในขณะที่ [2 4 6 8] มี 0 คู่



exec(input().strip())
```

จำนวนช่องทั้งหมดของอาเรย์ A หาได้จาก A.size เช่น ถ้า A.shape มีค่า (3,4,5) A.size มีค่า 60

$$eq\left(\begin{bmatrix}1&2&3\\4&5&6\\7&8&9\end{bmatrix},\begin{bmatrix}1&2&3\\4&5&0\\7&0&9\end{bmatrix},70\right)$$ ได้ True

$$closest\_point\_indexes\left(\begin{bmatrix}1&0\\9&9\\-1&0\\0&1\end{bmatrix},\begin{bmatrix}0&0\end{bmatrix}\right)$$ ได้ [0 2 3]

ข้อมูลนำเข้า

คำสั่งภาษา Python ที่ใช้ทดสอบการทำงานของฟังก์ชัน

ข้อมูลส่งออก

ผลที่ได้จากการสั่งทำงานคำสั่งที่ได้รับ

ตัวอย่าง

input (จากแป้นพิมพ์)	output
`print(eq(np.array([1,2]),np.array([1,0]),50))`	`True`
`print(eq(np.array([[[1,2]]]),np.array([[[1,1]]]),51))`	`False`
`print(closest_point_indexes(np.array([[9,9],[1,2]]),np.array([1,1])))`	`[1]`
`print(closest_point_indexes(np.array([[0,3],[3,0]]),np.array([1,1])))`	`[0 1]`
`print(number_of_inversions(np.array([1,2,3,4,5])))`	`0`
`print(number_of_inversions(np.array([5,4,3,2,1])))`	`10`
`print(number_of_inversions(np.array([1,3,5,2,4,6])))`	`3`

P-30: การเขียน slice จากลำดับ indexes

จงเขียนโปรแกรมรับลำดับของอินเด็กซ์ (ที่ไม่ซ้ำกัน เรียงจากน้อยไปมาก) เพื่อแปลงเป็นลำดับของสไลซ์และอินเด็กซ์ โดยรองรับเฉพาะรูปแบบดังนี้

- ลำดับของอินเด็กซ์ที่ติดกันและเพิ่มขึ้นทีละหนึ่ง จะถูกแปลงให้เป็นสไลซ์ในรูปแบบ **`start:stop`** เช่น **`2 3 4 5`** แปลงเป็น **`2:6`**
- อินเด็กซ์ที่ไม่ไปเป็นตามข้อที่แล้ว ให้คงไว้เหมือนเดิม เช่น **`2 3 4 5 9 11 13 14`** แปลงเป็น **`2:6 9 11 13:15`**

ข้อมูลนำเข้า

ลำดับของอินเด็กซ์ซึ่งเป็นเลขจำนวนเต็มไม่ติดลบ เรียงจากน้อยไปมาก

ข้อมูลส่งออก

ลำดับของสไลซ์และอินเด็กซ์ในรูปแบบที่อธิบายข้างต้น

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
`99`	`99`
`10 11 12 13 14`	`10:15`
`1 3 5 7 9`	`1 3 5 7 9`
`1 2 3 4 5 100`	`1:6 100`
`1 2 5 6 9 10`	`1:3 5:7 9:11`

P-31: การเขียน slice จากลำดับ indexes (อีกแบบ)

จงเขียนโปรแกรมรับลำดับของอินเด็กซ์ (ที่ไม่ซ้ำกัน เรียงจากน้อยไปมาก) เพื่อแปลงเป็นลำดับของสไลซ์ ตามรูปแบบดังนี้

- ลำดับของอินเด็กซ์ที่ติดกันและเพิ่มขึ้นทีละ **k** จะถูกแปลงให้เป็นสไลซ์ในรูปแบบ **start:stop:k** (โดย **stop** จะมีค่าเกินตัวสุดท้ายของสไลซ์อยู่หนึ่ง) เช่น **2, 4, 6, 10, 20, 31** แปลงเป็น **2:7:2, 10:21:10, 31:32:1**
- การพิจารณาจะทำจากซ้ายไปขวา หาสไลซ์ที่ยาวสุด ๆ เท่าที่จะได้จากซ้ายไปขวา

  เช่น **2, 4, 17, 20, 30** จะแปลงเป็น **2:5:2, 17:21:3, 30:31:1** ไม่ใช่ **2:5:2, 17:18,1, 20:31:10**

ข้อมูลนำเข้า

ลำดับของอินเด็กซ์ซึ่งเป็นเลขจำนวนเต็มไม่ติดลบ เรียงจากน้อยไปมาก คั่นด้วยจุลภาคและช่องว่าง

ข้อมูลส่งออก

ลำดับของสไลซ์ในรูปแบบที่อธิบายข้างต้น คั่นแต่ละสไลซ์ด้วยจุลภาคและช่องว่าง

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
99	99:100:1
10, 11, 12, 13, 14	10:15:1
1, 3, 5, 7, 9	1:10:2
0, 10, 11, 12, 13, 14, 15, 19	0:11:10, 11:16:1, 19:20:1
1, 2, 5, 7, 9, 10, 13	1:3:1, 5:10:2, 10:14:3

P-32: ป้ายทะเบียนรถ

รหัสป้ายทะเบียนยานพาหนะของประเทศเล็ก ๆ แห่งหนึ่ง ประกอบด้วยตัวเลข **1** ตัว ตัวอักษรภาษาอังกฤษตัวใหญ่ **2** ตัว และตัวเลขอีก **3** ตัว เช่น **9RU-432** หรือ **0XY-009** เป็นต้น กรมขนส่งฯ จะออกรหัสป้ายทะเบียนเรียงไปเรื่อย ๆ เริ่มตั้งแต่ป้ายแรกของประเทศคือ **0AA-000** ป้ายถัดไปก็คือ **0AA-001** ... ป้ายสุดท้ายคือ **9ZZ-999**

จงเขียนโปรแกรมที่รับรหัสป้ายทะเบียนมาหนึ่งรหัส และจำนวนเต็ม **m** เพื่อหาว่า ถัดจากรหัสป้ายทะเบียนที่ได้รับไปอีก **m** ป้าย จะเป็นรหัสอะไร เช่น ถัดจาก **2ZZ-998** อีก **2** ป้าย ก็คือ **3AA-000**

ข้อมูลนำเข้า

บรรทัดแรกเป็นรหัสป้ายทะเบียน ซึ่งเป็นสตริงในรูปแบบ ***NAA-NNN*** โดยที่ ***N*** คือเลข **0** ถึง **9** และ ***A*** คือ ตัวอักษรภาษาอังกฤษ **A** ถึง **Z**

บรรทัดที่สองเป็นจำนวนเต็ม **m**

ข้อมูลส่งออก

รหัสป้ายทะเบียนที่อยู่ถัดจากรหัสที่ได้รับไปอีก **m** ป้าย

ตัวอย่าง

input (จากแป้นพิมพ์)	**output** (ทางจอภาพ)
0AA-000 **899**	**0AA-899**
2ZZ-998 **2**	**3AA-000**
8XY-900 **222**	**8XZ-122**
4XY-999 **3000000**	**9JI-999**

ข้อแนะนำ

- ถ้าอยากรู้ว่า **"Z"** เป็นตัวที่เท่าไรในลำดับตัวอักษรภาษาอังกฤษ ก็ใช้คำสั่ง **ord("Z") - ord("A") + 1** จะได้ค่า **26**
- ดังนั้น ถ้าอยากรู้ว่าตัวอักษรในตัวแปร **x** (ซึ่งต้องมีตัวอักษรตัวเดียว) เป็นตัวที่เท่าไรในลำดับตัวอักษรภาษาอังกฤษ ก็ใช้คำสั่ง **ord(x) - ord("A") + 1**

P-33: การอ่านจำนวนเต็มขนาดใหญ่

จงเขียนโปรแกรมที่รับจำนวนเต็มไม่ติดลบ เพื่อแสดงคำอ่านของจำนวนเต็มนี้ในภาษาอังกฤษ (ดูตัวอย่าง)

ข้อมูลนำเข้า

จำนวนเต็ม **N** โดยที่ (**0 ≤ N ≤ 999999999999999**)

ข้อมูลส่งออก

ข้อความแทนคำอ่านในภาษาอังกฤษของจำนวนเต็มที่รับมา

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
0	zero
9012	nine thousand twelve
4003002001	four billion three million two thousand one
1234567890	one billion two hundred thirty four million five hundred sixty seven thousand eight hundred ninety
999999999999999	nine hundred ninety nine trillion nine hundred ninety nine billion nine hundred ninety nine million nine hundred ninety nine thousand nine hundred ninety nine

คำศัพท์

zero, one, two, three, four, five, six, seven, eight, nine,

ten, eleven, twelve, thirteen, fourteen, fifteen, sixteen, seventeen, eighteen, nineteen

twenty, thirty, forty, fifty, sixty, seventy, eighty, ninety

hundred, thousand, million, billion, trillion

P-34: การเขียนจำนวนเต็มขนาดใหญ่จากคำอ่าน

จงเขียนโปรแกรมที่รับคำอ่านของจำนวนเต็มในภาษาอังกฤษ เพื่อแสดงจำนวนเต็มที่แทนคำอ่านนั้น (ดูตัวอย่าง)

ข้อมูลนำเข้า

ข้อความแทนคำอ่านในภาษาอังกฤษของจำนวนเต็ม (ไม่ติดลบ) (จำนวนเต็มนี้มีค่าไม่เกิน $10^{15} - 1$)

ข้อมูลส่งออก

จำนวนเต็มที่แทนคำอ่านที่รับเข้ามา

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
`zero`	`0`
`nine thousand twelve`	`9012`
`four billion three million two thousand one`	`4003002001`
`one billion two hundred thirty four million five hundred sixty seven thousand eight hundred ninety`	`1234567890`
`nine hundred ninety nine trillion nine hundred ninety nine billion nine hundred ninety nine million nine hundred ninety nine thousand nine hundred ninety nine`	`999999999999999`

คำศัพท์

`zero, one, two, three, four, five, six, seven, eight, nine,`

`ten, eleven, twelve, thirteen, fourteen, fifteen, sixteen, seventeen, eighteen, nineteen`

`twenty, thirty, forty, fifty, sixty, seventy, eighty, ninety`

`hundred, thousand, million, billion, trillion`

P-35: การแปลงเศษส่วนเป็นจำนวนที่มีจุดทศนิยม

เราสามารถเขียนจำนวนตรรกยะในรูปแบบเศษส่วนหรือแบบทศนิยมได้ เช่น $\frac{1}{8} = 0.125$ แต่ก็มีจำนวนตรรกยะที่เขียนออกมาได้เป็นเลขหลังจุดทศนิยมไม่รู้จบแบบซ้ำ เช่น $\frac{3221}{555} = 5.8036036036036036...$ (เลข 036 จะซ้ำไปเรื่อย ๆ ไม่รู้จบ) ในกรณีนี้ ขอเขียนเป็น 5.8(036) แสดงให้เห็นว่า เลขในวงเล็บ 036 จะซ้ำไม่รู้จบ จงเขียนโปรแกรมที่รับจำนวนในรูปแบบเศษส่วน เพื่อแสดงจำนวนนี้ในรูปแบบทศนิยม

ข้อมูลนำเข้า

บรรทัดแรกเป็นจำนวนเต็มไม่ติดลบ **n**

บรรทัดที่สองเป็นจำนวนเต็มไม่ติดลบ **d**

n กับ **d** แทนเศษส่วน $\frac{n}{d}$ โดยที่ $0 \leq n \leq 1000000$ และ $1 \leq d \leq 1000000$

ข้อมูลส่งออก

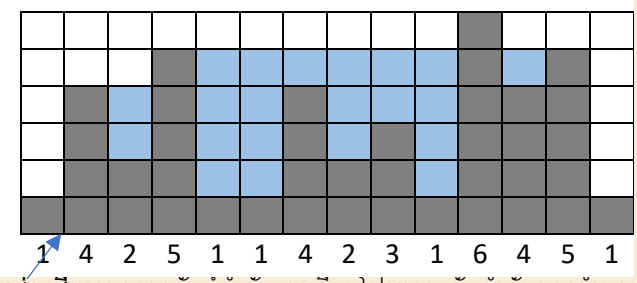
จำนวนในรูปแบบทศนิยมของเศษส่วนที่รับมา

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
1 2	0.5
1 3	0.(3)
1 12	0.08(3)
1 44	0.02(27)
1 591	0.(0016920473773265651438240270727580372250423011844331641285956006768189509306260575296108291032148 9)
125 3	41.(6)
125 5	25.
0 5	0.

P-36: น้ำรอระบาย

ข้างล่างนี้เป็นภาคตัดขวางของพื้นถนน สีเทาคือปูนและพื้นถนน ถ้าฝนตก น้ำจะขัง แสดงด้วยสีฟ้าในรูป



แต่ละคอลัมน์ในรูปข้างบนนี้มี**จำนวนช่องสีเทา**ของคอลัมน์กำกับ จงเขียนโปรแกรมรับลำดับของจำนวนเหล่านี้ และแสดงพื้นที่ของบริเวณช่องสีฟ้า

เพื่อความง่าย กำหนดให้สี่เหลี่ยมย่อย ๆ ในรูปมีขนาด **1×1** (ตัวอย่างข้างบนนี้มีจำนวนช่องฟ้าเท่ากับ **21** ช่อง เป็นคำตอบ)

ข้อมูลนำเข้า

รายการของจำนวนเต็มบวกที่แทนจำนวนช่องสีเทาในแต่ละคอลัมน์ จากซ้ายไปขวา คั่นด้วยช่องว่าง (ข้อมูลทดสอบมีจำนวนคอลัมน์ไม่เกิน **50**)

ข้อมูลส่งออก

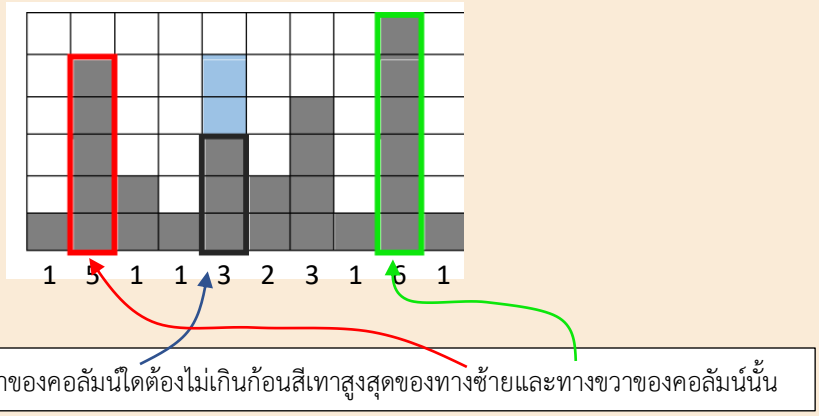
จำนวนช่องสีฟ้า ที่คำนวณจากข้อมูลที่ได้รับ

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
1 1 1 1 1	0
1 2 3 4 5 6 7 8 9 10	0
5 1 5	4
5 1 10	4
1 10 1 5 1	4
10 5 4 3 2 1 2 3 4 5 6 7 8 9	58
1 4 2 5 1 1 4 2 3 1 6 4 5 1	21

คำแนะนำ

- สามารถหาจำนวนก้อนสีฟ้าที่อยู่เหนือแต่ละคอลัมน์ของก้อนสีเทาได้ง่าย ๆ โดยสังเกตจากรูปข้างล่างนี้



P-37: การแยกคำออกจากตัวพิมพ์หลังอูฐ

Camel case คือการนำคำในวลีหรือประโยคภาษาอังกฤษมาเขียนติดกัน โดย

- ให้เปลี่ยนให้ตัวอักษรตัวแรกของคำเป็นตัวใหญ่ ส่วนตัวอื่นของคำเป็นตัวเล็ก ยกเว้น
- คำแรก และ คำที่ตามหลังตัวเลข ให้คงตัวอักษรตัวแรกเหมือนเดิม ส่วนตัวอื่นของคำให้เป็นตัวเล็ก

เช่น

- **Happy new year 2020 ja** ได้ camel case เป็น **HappyNewYear2020ja**
- **happy new year 2020 Ja** ได้ camel case เป็น **happyNewYear2020Ja**

โจทย์ข้อนี้ไม่ได้ให้นำคำต่าง ๆ มาประกอบเป็น camel case แต่ให้แยกสตริงที่เป็น camel case ออกเป็นคำ ๆ

ข้อมูลนำเข้า

สตริงแบบ camel case ที่มีแต่ตัวอักษรภาษาอังกฤษกับตัวเลข

ข้อมูลส่งออก

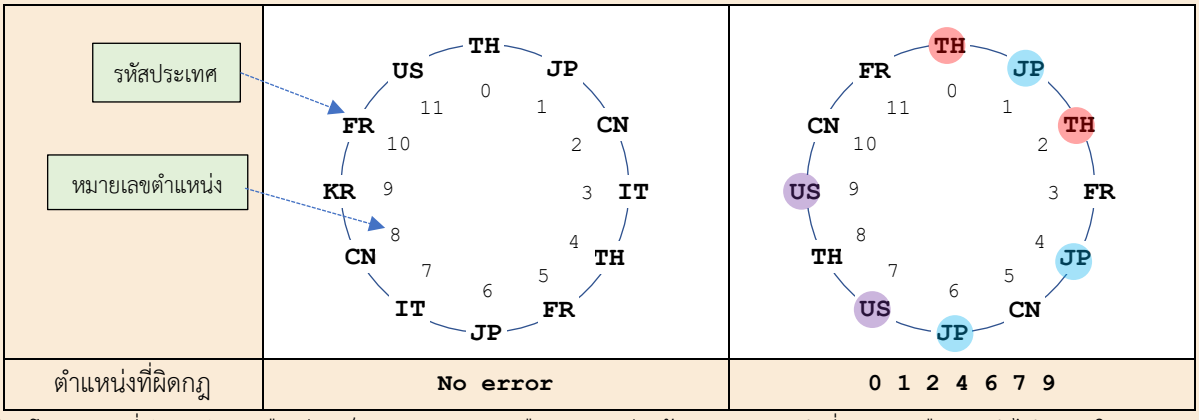
คำต่าง ๆ ที่ได้จากการแยกสตริงที่ได้รับมา แสดงบนบรรทัดเดียว คั่นแต่ละคำด้วยช่องว่าง ตามลำดับที่ปรากฏในสตริงที่รับมา (ดูตัวอย่างประกอบ)

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
HappyNewYear2020	Happy New Year 2020
happyBirthDay2u	happy Birth Day 2 u
h20m10s15	h 20 m 10 s 15
ABBADancingQueen16August1976	A B B A Dancing Queen 16 August 1976

P-38: ใกล้กันเกิน

ในงานลูกเสือเนตรนารีโลกครั้งหนึ่งมีผู้ร่วมงานมากมายจากหลายประเทศ ฐานกิจกรรมหนึ่งจัดให้ลูกเสือเนตรนารีจำนวนหนึ่งมายืนเรียงเป็นวงกลม โดยมีกฎว่า ลูกเสือเนตรนารีสองคนใดที่มาจากประเทศเดียวกันต้องยืนห่างกันโดยมีประเทศอื่นคั่นอย่างน้อยสามคน เช่น การยืนเรียงในรูปซ้ายข้างล่างนี้ถูกกฎ ในขณะที่รูปทางขวาผิดกฎ (อักษรย่อที่วงนอกในรูปคือ รหัสประเทศ ส่วนที่อยู่วงในคือหมายเลขตำแหน่งที่ยืน)



ตำแหน่งที่ผิดกฎ	No error	0 1 2 4 6 7 9

จงเขียนโปรแกรมที่รับลำดับการยืนเรียงเป็นวงกลมของลูกเสือเนตรนารี แล้วแสดงตำแหน่งที่ผิดกฎ หรือสรุปว่าไม่ผิดกฎใด ๆ เลย

ข้อมูลนำเข้า

บรรทัดเดียวเป็นรายการของรหัสประเทศคั่นด้วยช่องว่าง เริ่มต้นซ้ายสุดที่ตำแหน่งที่ `0` เป็นต้นไป

ข้อมูลส่งออก

ถ้ารูปแบบการยืนเรียงผิดกฎ ให้แสดงตำแหน่งที่ผิด จากตำแหน่งน้อยไปมาก ตำแหน่งละบรรทัด ถ้าไม่ผิดกฎเลย ให้แสดง `No error`

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
`TH JP US CN`	`No error`
`TH FR TH TH`	`0` `2` `3`
`TH JP TH FR JP CN JP US TH US CN FR`	`0` `1` `2` `4` `6` `7` `9`

P-39: รหัส Baconian

นาย ก กับนาย ข สื่อสารตัวเลขลับ ๆ กันตลอด ทั้งคู่จึงตกลงวิธีการเข้าและถอดรหัสที่รู้กันสองคน ถ้า ก ต้องการส่งหมายเลขโทรศัพท์ รปภ. จุฬาฯ **02-218-000** ให้ ข ก็จะส่งข้อความ

HAPPY BiRtHdAY To YOU !!HapPY BiRtHDAY TO YOU !!HAPPY BIR

เมื่อ ข ได้รับ ก็เริ่มถอดรหัสโดย 1) เลือกดูเฉพาะตัวอักษรอังกฤษ 2) ดูว่าตัวไหนเป็นตัวพิมพ์ใหญ่หรือเล็ก 3) ดูทีละ 4 ตัวที่ติดกัน และ 4) ใช้ตารางทางขวานี้แปลงทีละ 4 ตัวที่ติดกันเป็นตัวเลข ดังตัวอย่างนี้

รหัส	ข้อมูลจริง
UUUU	0
UUUL	1
UULU	2
UULL	3
ULUU	4
ULUL	5
ULLU	6
ULLL	7
LUUU	8
LUUL	9
LULU	-
LULL	,

L แทนตัวพิมพ์เล็ก

U แทนตัวพิมพ์ใหญ่

จากข้อความที่ ข ได้รับ

(1) ดูเฉพาะตัวอักษร **HAPPYBiRtHdAYToYOUHapPYBiRtHDAYTOYOUHAPPYBIR**

(2) เป็นพิมพ์ใหญ่หรือเล็ก **UUUUUULULULUUULUUUULLUUULULUUUUUUUUUUUUUUUUUU**

(3) ดูทีละ **4** ตัวที่ติดกัน **UUUUULULULULUULUUULLUUULULUUUUUUUUUUUUUUUU**

(4) แปลงจากตารางทางขวา **0 2 - 2 1 8 - 0 0 0 0**

จะว่าไปแล้ว ก จะส่งข้อความ **OOOOOOoOoOoOOOoOOOOooOOOoOoOOOOOOOOOOOOOOOO** ให้ ข ก็ได้ เมื่อถอดรหัสด้วยวิธีข้างบนนี้ก็ได้หมายเลขโทรศัพท์เดียวกัน

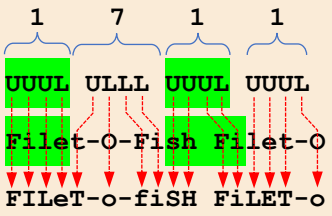
สำหรับขั้นตอนเข้ารหัส สิ่งที่ผู้ส่งต้องทำ คือ เปลี่ยนชุดตัวเลข (อาจมีเครื่องหมาย **-** กับ **,** ด้วยก็ได้) โดยอาศัยข้อความหลอกประกอบการเข้ารหัส เช่น ต้องการส่งเลข **1711** โดยใช้ข้อความหลอกคือ **Filet-O-Fish** เนื่องจากเลขที่ต้องการส่งมี 4 ตัว ต้องใช้ตัวอักษรอังกฤษ 4x4 = 16 ตัว แต่ข้อความหลอก **Filet-O-Fish** มีตัวอักษรอังกฤษแค่ 10 ตัว จึงต้องนำข้อความหลอกมาต่อกันหลายชุด เพื่อให้ได้จำนวนตัวอักษรอังกฤษตามที่ต้องการ ดังตัวอย่างข้างล่างนี้

(1) พิจารณาเลขที่ต้องการส่ง

(2) แปลงเลขเป็นรหัสพิมพ์ใหญ่/เล็กตามตารางข้างบน

(3) นำข้อความหลอกมาต่อกันหลายชุด

(4) เปลี่ยนตัวอักษรใน (3) ให้เป็นตัวพิมพ์ใหญ่/เล็กตาม (2)



จงเขียนโปรแกรมเข้ารหัสหรือถอดรหัส ด้วยวิธีข้างต้น

ข้อมูลนำเข้า

บรรทัดแรกเป็นข้อความหลอก

บรรทัดที่สองเป็นจำนวนเต็ม บอกจำนวนบรรทัดที่จะตามมา แต่ละบรรทัดเป็นการเข้าหรือถอดรหัสที่ต้องทำ

บรรทัดต่อมา แต่ละบรรทัดอาจอยู่ในรูปแบบใดแบบหนึ่ง ข้างล่างนี้

- **E** ตามด้วย ชุดตัวเลข **0** ถึง **9** อาจมีเครื่องหมาย **-** กับ **,** ที่ต้องการให้เข้ารหัส โดยใช้ข้อความหลอกที่กำหนดในบรรทัดแรก
- **D** ตามด้วย ข้อความที่ต้องการให้ถอดรหัส

ข้อมูลส่งออก

ผลลัพธ์ของการเข้าหรือถอดรหัส ที่กำหนดในข้อมูลขาเข้า

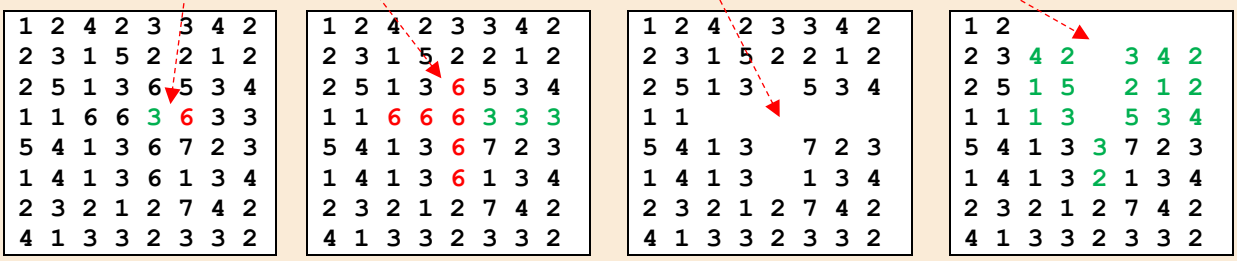
ตัวอย่าง

Input (จากแป้นพิมพ์)	**Output** (ทางจอภาพ)
O 2 E 218-0000 D THe QUICk bROWn FoX JUMPS OVER THE LAZY	OOoOOOOoOOOOoOoOOOOOOOOOOOOOOOO 218-0000
Lotto_Winner_ 5 E 34-20 E 19-50 E 27-125 E 54-100 E 968-50	LOttO_wINnEr_LOTtO_WINN LOTto_WInnEr_LOtTo_WINN LOtTO_winnEr_LOTTo_WInNEr_Lo LoTtO_wINnEr_LOTTo_WINNER_LO lOTtO_wiNnER_LoTtO_WiNnER_LO
Baconian 5 D LOttO wINnEr LOTtO WINN D LOTto WInnEr LOtTo WINN D LOtTO winnEr LOTTo WInNEr Lo D LoTtO wINnEr LOTTo WINNER LO D lOTtO wiNnER LoTtO WiNnER LO	34-20 19-50 27-125 54-100 968-50
midterm-exam- 8 E 1,2,1,3,2 E 2,4,3,2,4 E 2,3,2,1,1 E 2,3,2,3,4 D MIDteRm-eXAm-MiDteRM-ExaM-miDTerm-ExaM-MiD D MIdTeRm-eXaM-MiDteRM-exaM-miDTeRm-ExaM-mID D MIdTeRm-eXAm-miDteRM-eXaM-miDTErm-ExaM-MId D MIdTeRm-eXAm-miDteRM-eXaM-miDTerm-ExaM-mID	MIDteRm-eXAm-MiDteRM-ExaM-miDTerm-ExaM-MiD MIdTeRm-eXaM-MiDteRM-exaM-miDTeRm-ExaM-mID MIdTeRm-eXAm-miDteRM-eXaM-miDTErm-ExaM-MId MIdTeRm-eXAm-miDteRM-eXaM-miDTerm-ExaM-mID 1,2,1,3,2 2,4,3,2,4 2,3,2,1,1 2,3,2,3,4

P-40: เกม Bejeweled

เกม Bejeweled แบบดั้งเดิมมีตารางแบ่งเป็นช่อง ๆ ขนาด 8x8 แต่ละช่องมีพลอย (ซึ่งมี 7 แบบ) วางอยู่ดังรูปขวา ผู้เล่นต้องเลือกสลับพลอย 2 เม็ดที่อยู่ติดกัน (ทำได้ครั้งละคู่) เพื่อให้พลอย 3 เม็ดขึ้นไป ที่เรียงติดกันเป็นชนิดเดียวกัน จะทำให้พลอยที่ติดกันเหมือนกันหายไป (และผู้เล่นก็ได้คะแนน) เช่น สลับคู่นี้ เมื่อพลอยหาย พลอยเม็ดบน ๆ จะหล่นลงมาแทน ก็จะมีพลอยเม็ดใหม่มาเติมแถวบน ๆ ผู้เล่นก็เลือกต่อ

สำหรับโจทย์ข้อนี้ ขอแทนเม็ดพลอยแต่ละชนิดด้วยตัวเลข ดูตัวอย่างข้างล่างนี้ เริ่มจากตารางซ้ายสุด ถ้าผู้เล่นเลือกสลับ เลข **3** กับ **6** จะทำตารางเปลี่ยนไปเป็นตารางทางขวาถัดมา จะพบว่ามีเลขที่ติดกันเหมือนกันสามตัวขึ้นไป ดังแสดงในรูป (คือเลข **3** กับ **6**) ทำให้เลขเหล่านี้หายไป แล้วเลขในแถวบน ๆ ก็หล่นลงมา



ในกรณีผู้ใช้สลับตัวเลขที่ติดแล้ว ไม่เกิดเลขติดกันเหมือนกันสามตัวขึ้นไป ก็จะไม่มีอะไรเปลี่ยนแปลง (ที่สลับไปจะสลับกลับเหมือนเดิม)

จงเขียนโปรแกรมรับตารางตัวเลข จากนั้นรับตำแหน่งของเลขที่ติดกันที่ต้องการสลับกัน แล้วก็แสดงผลที่เกิดขึ้น

ข้อมูลนำเข้า

แปดบรรทัดแรกเป็นลักษณะของตาราง บรรทัดละแถว ประกอบด้วยเลข 8 ตัว

บรรทัดต่อมามีจำนวนเต็ม 4 จำนวน ระบุ เลขแถว เลขคอลัมน์ของตัวเลขสองตัวที่ติดกันในตารางที่ต้องการสลับ (ตำแหน่งตรงนี้เป็นตำแหน่งที่ติดกันแน่ ๆ โดยเลขตำแหน่งของแถวบนสุด และคอลัมน์ซ้ายสุด คือ 0)

ตัวเลข **1**, **2** หรือ **3** ระบุผลลัพธ์ที่ต้องการให้แสดง (ตัวเลขตรงนี้เป็นเลข 1, 2 หรือ 3 แน่ ๆ)

ข้อมูลส่งออก

ผลลัพธ์ที่แสดง ขึ้นกับตัวเลขในบรรทัดสุดท้ายของอินพุต

- ถ้าเป็นเลข **1** : แสดงจำนวนตัวเลขทั้งหมดในตารางที่จะหายไป
- ถ้าเป็นเลข **2** : แสดงตัวเลขในตาราง หลังจากลบตัวเลขที่หายไปแล้ว
- ถ้าเป็นเลข **3** : แสดงตัวเลขในตาราง หลังจากตัวเลขแถวบน ๆ หล่นลงมาแทนตัวเลขที่หายไปในตาราง

ตัวอย่าง

Input (จากแป้นพิมพ์)	**Output** (ทางจอภาพ)
1 2 4 2 3 3 4 2 2 3 1 5 2 2 1 2 2 5 1 3 6 5 3 4 1 1 6 6 3 6 3 3 5 4 1 3 6 7 2 3 1 4 1 3 6 1 3 4 2 3 2 1 2 7 4 2 4 1 3 3 2 3 3 2 3 4 3 5 1	9

1 2 4 2 3 3 4 2 2 3 1 5 2 2 1 2 2 5 1 3 6 5 3 4 1 1 6 6 3 6 3 3 5 4 1 3 6 7 2 3 1 4 1 3 6 1 3 4 2 3 2 1 2 7 4 2 4 1 3 3 2 3 3 2 3 4 3 5 2	1 2 4 2 3 3 4 2 2 3 1 5 2 2 1 2 2 5 1 3 5 3 4 1 1 5 4 1 3 7 2 3 1 4 1 3 1 3 4 2 3 2 1 2 7 4 2 4 1 3 3 2 3 3 2
1 2 4 2 3 3 4 2 2 3 1 5 2 2 1 2 2 5 1 3 6 5 3 4 1 1 6 6 3 6 3 3 5 4 1 3 6 7 2 3 1 4 1 3 6 1 3 4 2 3 2 1 2 7 4 2 4 1 3 3 2 3 3 2 3 4 3 5 3	1 2 2 3 4 2 3 4 2 2 5 1 5 2 1 2 1 1 1 3 5 3 4 5 4 1 3 3 7 2 3 1 4 1 3 2 1 3 4 2 3 2 1 2 7 4 2 4 1 3 3 2 3 3 2
1 2 4 2 3 3 4 2 2 3 1 5 2 2 1 2 2 5 1 3 6 5 3 4 1 1 6 6 3 6 3 3 5 4 1 3 6 7 2 3 1 4 1 3 6 1 3 4 2 3 2 1 2 7 4 2 4 1 3 3 2 3 3 2 3 4 3 3 2	1 2 4 2 3 3 4 2 2 3 1 5 2 2 1 2 2 5 1 5 3 4 1 1 6 6 3 3 5 4 1 7 2 3 1 4 1 1 3 4 2 3 2 1 2 7 4 2 4 1 3 3 2 3 3 2
1 2 4 2 3 3 4 2 2 3 1 5 2 2 1 2 2 5 1 3 6 5 3 4 1 1 6 6 3 6 3 3 5 4 1 3 6 7 2 3 1 4 1 3 6 1 3 4 2 3 2 1 2 7 4 2 4 1 3 3 2 3 3 2 3 4 3 3 3	1 2 4 3 4 2 2 3 1 2 1 2 2 5 1 5 3 4 1 1 6 6 3 3 5 4 1 2 3 7 2 3 1 4 1 5 2 1 3 4 2 3 2 1 2 7 4 2 4 1 3 3 2 3 3 2
1 2 4 2 3 3 4 2 2 3 1 5 2 2 1 2 2 5 1 3 6 5 3 4 1 1 6 6 3 6 3 3 5 4 1 3 6 7 2 3 1 4 1 3 6 1 3 4 2 3 2 1 2 7 4 2 4 1 3 3 2 3 3 2 6 6 7 6 1	0

P-41: เกม BINGO

BINGO เป็นเกมมีหลายแบบ สำหรับในโจทย์นี้ ผู้เล่นแต่ละคนจะมีแผ่นบิงโกสี่เหลี่ยม ภายในแบ่งเป็นตาราง **5×5** แต่ละช่องมีตัวเลขสุ่มที่เป็นไปได้ตั้งแต่ **1** ถึง **75** บนสุดของตารางเขียนตัวอักษร **B I N G O** กำกับแต่ละคอลัมน์ เมื่อเริ่มเล่นเกม ผู้ดำเนินเกมที่เรียกว่า **caller** จะสุ่มหยิบลูกปิงปอง แสดงรหัสที่มีตัวอักษรหนึ่งตัว (จากคำว่า **BINGO**) และตัวเลข (**1** ถึง **75**) ถ้าผู้เล่นคนใดพบว่า แผ่นบิงโกของตนมีรหัสกับที่ **caller** สุ่มขึ้นมา ก็ทำเครื่องหมายที่ช่องบนแผ่นบิงโกของตน เช่น ถ้า **caller** หยิบได้ **B6** จะตรงกับช่องที่แถวสี่ คอลัมน์ **B** ของแผ่นบิงโกทางขวานี้ เมื่อใดรหัสที่ช่องบนแผ่นบิงโกตรงกับลูกปิงปองที่สุ่มขึ้นมาในแถวแนวนอนทั้งแถว หรือในแถวแนวตั้งทั้งแถว หรือในแถวแนวทแยงทั้งแถว (จากซ้ายบนมาขวาล่าง หรือจากขวาบนมาซ้ายล่าง) เราก็จะตะโกนคำว่า "บิงโก" ตามด้วยรหัสต่าง ๆ ที่อยู่ในแนวบิงโก เพื่อรับรางวัล ให้สังเกตว่า ช่องตรงกลางแผ่นบิงโกที่เขียนว่า ***** เป็นช่องได้ฟรี

B	I	N	G	O
3	13	23	5	29
10	22	31	17	70
1	4	*	27	33
6	65	48	29	15
11	70	62	2	18

จงเขียนโปรแกรมอ่านเลขบนแผ่นบิงโกหนึ่งแผ่น จากนั้นอ่านลำดับของลูกปิงปองที่ **caller** สุ่มขึ้นมา เมื่อใดที่พบว่า เราชนะบิงโกแล้ว ก็ให้แสดงจำนวนลูกปิงปองที่ **caller** หยิบจนเราชนะ พร้อมกับแสดงลำดับของรหัสในแนวที่ได้บิงโก

ข้อมูลนำเข้า

บรรทัดแรกเป็นสตริง **B I N G O**

ห้าบรรทัดต่อมาเป็นตัวเลขในแต่ละช่องของแผ่นบิงโกขนาด **5×5**

บรรทัดต่อ ๆ มา เป็นลำดับของรหัส (ตัวอักษรและเลข) ของลูกปิงปองที่ **caller** หยิบขึ้นมา บรรทัดละกี่ตัวก็ได้ คั่นด้วยช่องว่าง

ข้อมูลส่งออก

บรรทัดแรกเป็นจำนวนลูกปิงปองที่ **caller** หยิบจนกระทั่ง แผ่นบิงโกที่รับมาจะชนะ (ให้ถือว่าข้อมูลที่ได้รับมา จะทำให้ชนะแน่ ๆ)

บรรทัดต่อมา (มีได้มากสุดสามบรรทัด) แสดงลำดับของรหัสของลูกปิงปองที่อยู่ในแนวบิงโก โดยแสดงตามลำดับดังนี้

- ลำดับของรหัสในแถวแนวนอน (ถ้ามี) จากซ้ายไปขวา
- ลำดับของรหัสในแถวแนวตั้ง (ถ้ามี) จากบนลงล่าง
- ลำดับของรหัสในแนวทแยงมุม (ถ้ามี) จากซ้ายบนลงมาขวาล่าง
- ลำดับของรหัสในแนวทแยงมุม (ถ้ามี) จากซ้ายล่างขึ้นไปมาขวาบน
- ให้สังเกตว่า จะเกิดบิงโกในแนวทแยงมุมได้แค่แนวเดียว และไม่ต้องแสดง ***** ที่ช่องกลางของตารางบิงโก

ตัวอย่าง

Input (จากแป้นพิมพ์)

```
 B  I  N  G  O
 3 13 23  5 29
10 22 31 17 70
 1  4  * 27 33
 6 65 48 29 15
11 70 62  2 18
B1 O29 B2 G1 B3 B2 G10 N23 O28
G17 O70 I13
B1 G5 I4 G27 O33
```

Output (ทางจอภาพ)

```
14
B3, I13, N23, G5, O29
```

เลข **14** นี้มาจากจำนวนการอ่านลำดับ

```
B1 O29 B2 G1 B3 B2 G10 N23 O28
G17 O70 I13
B1 G5
```

พอถึง **G5** เป็นตัวที่ **14** ก็ได้บิงโก หยุดอ่านได้แล้ว

Input (จากแป้นพิมพ์)

```
 B  I  N  G  O
 3 13 23  5 29
10 22 31 17 70
 1  4  * 27 33
 6 65 48 29 15
11 70 62  2 18
I13 N23 G5 O29 B11 G70 N18
B6 B1 B10 O18 O29 B3 N62
```

Output (ทางจอภาพ)

```
13
B3, I13, N23, G5, O29
B3, B10, B1, B6, B11
```

B I N G O 3 13 23 5 29 10 22 31 17 70 1 4 * 27 33 6 65 48 29 15 11 70 62 2 18 I13 N23 G5 O29 B11 G70 N18 B6 B1 B10 O18 O29 O18 I22 G29 B3 I8 B70 N62	16 B3, I13, N23, G5, O29 B3, B10, B1, B6, B11 B3, I22, G29, O18 ไม่ต้องแสดง * ในแนวทแยงนี้
B I N G O 3 13 23 5 29 10 22 31 17 70 1 4 * 27 33 6 65 48 29 15 11 70 62 2 18 O2 O3 O4 B6 B1 B10 B3 I70 N62 G2 O18 O29 G17 I65 B9 N8 O7 G11 B11 B18 B19	19 B11, I70, N62, G2, O18 B3, B10, B1, B6, B11 B11, I65, G17, O29

P-42: เศษส่วนต่อเนื่อง (Continued Fraction) อย่างง่าย

จงเขียนโปรแกรมที่รับลำดับของจำนวนเต็ม A เพื่อคำนวณและแสดงลำดับของจำนวนจริง C ด้วยสูตรข้างล่างนี้

$$A = a_0, a_1, a_2, \cdots, a_{n-1}$$

$$C = a_0,\ a_0 + \frac{1}{a_1},\ a_0 + \frac{1}{a_1 + \frac{1}{a_2}}, \cdots,\ a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{\ddots\ a_{n-2} + \frac{1}{a_{n-1}}}}}$$

เช่น A = 1, 8, 7, 9 จะได้ $C = 1,\ 1 + \frac{1}{8},\ 1 + \frac{1}{8 + \frac{1}{7}},\ 1 + \frac{1}{8 + \frac{1}{7 + \frac{1}{9}}}$ = 1, 1.125, 1.122807, 1.122841

ข้อมูลนำเข้า

บรรทัดเดียวเป็นรายการของจำนวนเต็ม (คั่นด้วยช่องว่าง) แทนลำดับ A

ข้อมูลส่งออก

ข้อมูลใน C ที่คำนวณได้จาก A ที่ได้รับ ตัวละบรรทัด (A มีข้อมูล n ตัว C ก็มีข้อมูล n ตัว)

โดยแสดงเลขหลังจุดทศนิยมอย่างมาก **8** ตัว ด้วยฟังก์ชัน **`round(x, 8)`**

ตัวอย่าง

Input (จากแป้นพิมพ์)	**Output** (ทางจอภาพ)
`1 8 7 9`	`1` `1.125` `1.12280702` `1.12284069`
`1 1 1 1 1 1 1 1 1 1 1 1 1 1`	`1` `2.0` `1.5` `1.66666667` `1.6` `1.625` `1.61538462` `1.61904762` `1.61764706` `1.61818182` `1.61797753` `1.61805556` `1.61802575` `1.61803714`

P-43: การปรับแนวสตริงให้ตรงกันมากสุด

ให้ **X** และ **Y** เป็นสตริงของตัวอักษรภาษาอังกฤษ อยากหาว่า จะต้องเลื่อน **X** หรือ **Y** (ตัวใดตัวหนึ่ง) ไปทางขวาอย่างไร เพื่อให้ตัวอักษรใน **X** และ **Y** ที่ตำแหน่งตรงกันเหมือนกันเป็นจำนวนมากสุด เช่น

X = "ACGGCGGCTGG" กับ **Y = "GCGGAATGGCGTTTGCGAGAGCT"** พบว่า เลื่อน **X** ไปทางขวา **5** ตำแหน่ง จะทำให้มีตัวอักษรของทั้งคู่ตรงกันเหมือนกันมากสุด **7** ตัว	**-----ACGGCGGCTGG** **GCGGAATGGCGTTTGCGAGAGCT**
X = "AACTAAAGATCG" กับ **Y = " CAAAGTTCAACCC"** พบว่า เลื่อน **Y** ไปทางขวา **3** ตำแหน่ง จะทำให้มีตัวอักษรของทั้งคู่ตรงกันเหมือนกันมากสุด **6** ตัว	**AACTAAAGATCG** **---CAAAGTTCAACCC**

จงเขียนโปรแกรมรับสตริงสองชุด แล้วแสดงว่า ต้องเลื่อนตัวใดไปทางขวาเท่าใด เพื่อให้ได้ตัวที่ตรงกันเหมือนกันเป็นจำนวนมากสุด พร้อมทั้งแสดงจำนวนที่ตัวตรงกันเหมือนกันมากสุดนั้นด้วย ในกรณีที่เลื่อนได้หลายวิธีที่ได้ผลเหมือนกัน ให้แสดงเฉพาะอันที่เลื่อนด้วยจำนวนการเลื่อนน้อยสุด

ข้อมูลนำเข้า

สองบรรทัด บรรทัดละสตริง ประกอบด้วยตัวอักษรภาษาอังกฤษตัวพิมพ์ใหญ่เท่านั้น

ข้อมูลส่งออก

สองบรรทัดแรก แสดงผลการเลื่อนสตริงด้วยจำนวนการเลื่อนน้อยสุด เพื่อให้ได้จำนวนตัวอักษรที่ตรงกันเหมือนกันมากสุด (รูปแบบดังในตัวอย่าง) บรรทัดที่สาม แสดงจำนวนตัวอักษรที่ตรงกันเหมือนกันมากสุด

ตัวอย่าง

Input (จากแป้นพิมพ์)	**Output** (ทางจอภาพ)
ACGGCGGCTGG **GCGGAATGGCGTTTGCGAGAGCT**	**-----ACGGCGGCTGG** **GCGGAATGGCGTTTGCGAGAGCT** **7**
AACTAAAGATCG **CAAAGTTCAACCC**	**AACTAAAGATCG** **---CAAAGTTCAACCC** **6**
AAAAAAAAAA **AAAAAAA**	**AAAAAAAAAA** **AAAAAAA** **7**
AAAAAAAAA **GGGGGG**	**AAAAAAAAA** **GGGGGG** **0**

P-44: เลขนี้มีกี่หลัก

จงเขียนโปรแกรมรับค่า **M** และ **N** (โดยที่ **M** และ **N** คือจำนวนเต็มไม่ติดลบ **M ≤ N**) เพื่อหาจำนวนหลักของจำนวนเต็มที่ได้จากการนำจำนวนเต็มตั้งแต่ **M** ถึง **N** มาเขียนติดกัน เช่น **M = 9**, **N = 12** นำเลข **9,10,11,12** มาเขียนติดกันได้ **9101112** เป็นจำนวนเต็ม **7** หลัก

ข้อมูลนำเข้า

บรรทัดเดียวมีจำนวนเต็มสองจำนวน **M** กับ **N** โดยที่ **0 ≤ M ≤ N ≤ 99999999999999999999999999999** ข้อมูลทดสอบส่วนใหญ่จะมีค่า **M** และ **N** ที่ต่างกันมาก

ข้อมูลส่งออก

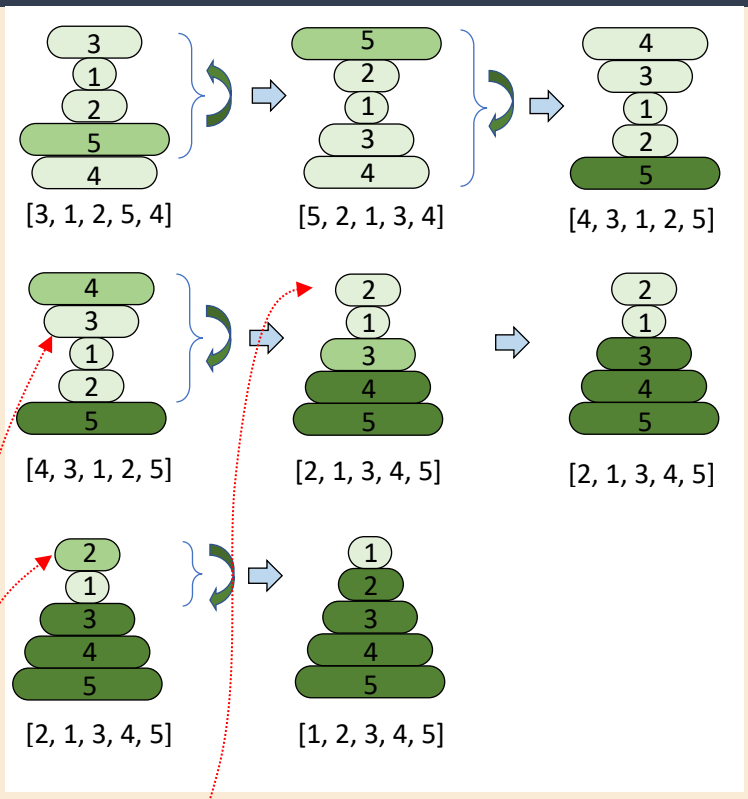
จำนวนหลักของจำนวนเต็มที่ได้จากการนำจำนวนเต็มตั้งแต่ **M** ถึง **N** มาเขียนติดกัน (รวม **M** กับ **N** ด้วย)

ตัวอย่าง

Input (จากแป้นพิมพ์)	**Output** (ทางจอภาพ)
0 9	10
1234567890 1234567890	10
1234 56789012345678901	954302098765426398
0 99999999999900000099999999	26888888888861888891588888890

P-45: การเรียงแพนเค้ก

สมมติว่า มีแพนเค็กจำนวนหนึ่งขนาดต่างกัน วางซ้อนกัน เราอยากจะเรียงแผ่นแพนเค็กให้ซ้อนทับกันให้เป็นระเบียบ ชิ้นเล็กวางข้างบนชิ้นใหญ่กว่าไปเรื่อย ๆ จากบนลงล่าง จะทำอย่างไร ? โดยอนุญาตให้ใช้เฉพาะวิธี*หยิบกองแพนเค็กที่ติดกัน แล้วพลิกแพนเค็กกองที่เลือกนั้น กลับด้านจากบนลงล่าง จากล่างขึ้นบน*



ขั้นตอนง่าย ๆ ก็คือ จากกองแพนเค็กที่ยังไม่ได้เรียง ให้หาว่าชิ้นใหญ่สุดอยู่ไหน หยิบแพนเค็กจากชิ้นบนสุดถึงชิ้นใหญ่สุดแล้วก็พลิก จะทำให้ชิ้นใหญ่สุดไปอยู่บนสุด จากนั้นก็หยิบทั้งกอง พลิกอีกครั้ง ชิ้นใหญ่สุดก็จะไปอยู่ล่างสุด อันเป็นตำแหน่งที่ชิ้นใหญ่นั้นควรอยู่ (ดูแถวบนของรูปด้านขวา) ใช้การพลิก 2 ครั้งเพื่อทำให้ชิ้นใหญ่สุดของกองมาอยู่ล่างสุด จากนั้นก็ทำเหมือนเดิม เพียงแต่ให้คิดเสียว่า ตอนนี้กองแพนเค็กลดขนาดลงหนึ่ง ไม่สนใจชิ้นใหญ่สุดที่อยู่ในที่ที่ควรอยู่แล้ว

แต่ก็มีกรณีที่ต้องพิจารณาเล็กน้อยเพื่อลดจำนวนการพลิก

- ถ้าชิ้นใหญ่สุดอยู่บนสุดอยู่แล้ว ก็พลิกครั้งเดียว
- ถ้าชิ้นใหญ่สุดของกองอยู่ด้านล่างสุดอยู่แล้ว ก็ไม่ต้องทำอะไร

ขอแทนขนาดและการจัดวางแพนเค็ก ด้วยลิสต์ของจำนวนเต็ม จำนวนเต็มในลิสต์แทนขนาด ซ้ายสุดของลิสต์คือแผ่นแพนเค็กบนสุดของกอง ไล่เรียงจากซ้ายไปขวาในลิสต์ ก็คือจากบนลงล่างในกองแพนเค็ก ขวาสุดชองลิสต์ก็คือล่างสุดของกอง

จงเขียนโปรแกรมรับรายการของจำนวนเต็ม (ที่แทนกองแพนเค็ก) เพื่อหาว่า รายการของจำนวนเต็มนี้จะเปลี่ยนแปลงอย่างไร จนทำให้เรียงลำดับเรียบร้อยตามต้องการด้วยวิธีที่อธิบายข้างต้น

ข้อมูลนำเข้า

บรรทัดเดียวเป็นรายการของจำนวนเต็ม (คั่นด้วยช่องว่าง)

ข้อมูลส่งออก

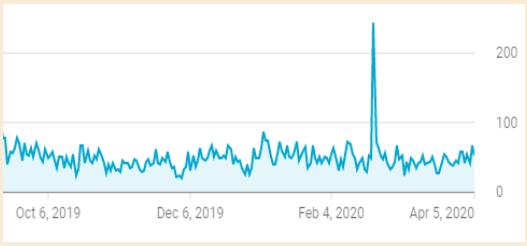
หลาย ๆ บรรทัด แสดงค่าในลิสต์ของจำนวนเต็มที่รับมา มีการเปลี่ยนแปลงค่าระหว่างการพลิกกองแพนเค็ก ด้วยวิธีข้างต้น เพื่อเรียงลำดับแพนเค็กให้ได้ตามที่ต้องการ

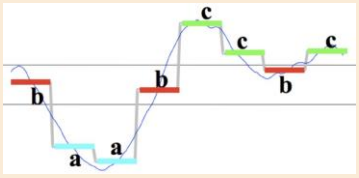
ตัวอย่าง

input (จากแป้นพิมพ์)	**output** (ทางจอภาพ)
`3 1 2 5 4`	`[3, 1, 2, 5, 4]` `[5, 2, 1, 3, 4]` `[4, 3, 1, 2, 5]` `[2, 1, 3, 4, 5]` `[1, 2, 3, 4, 5]`
`10 20 30 40 50 60`	`[10, 20, 30, 40, 50, 60]`

P-46: การแปลงอนุกรมเวลาเป็นสตริง (SAX)

อนุกรมเวลา (time series) คืออนุกรมของข้อมูลที่ลำดับแทนเวลา (รูปทางขวาเป็นเส้นกราฟที่วาดจากอนุกรมเวลาของยอดผู้เข้าชมวิดีโอบทแรกวิชา 2110101 ในช่วงเวลาหกเดือน) เราสามารถแปลงอนุกรมเวลาเป็นสตริงได้ด้วยวิธีหนึ่งที่ชื่อว่า SAX (Symbolic Aggregation approXimation)





SAX มีหลักการคร่าว ๆ คือ เริ่มด้วยการแปลงข้อมูลในอนุกรม **x** เป็นอนุกรมของคะแนนมาตรฐาน **z** จากนั้นสร้างอนุกรมใหม่ **p** ที่เก็บค่าเฉลี่ยของข้อมูลใน **z** ที่ติดกัน **k** ตัว ดังนั้นถ้าอนุกรมเริ่มต้น **x** มี **N** ตัว จะได้อนุกรมใหม่ **paa** มีข้อมูล **N/k** ตัว และขั้นตอนสุดท้ายคือแปลงค่าใน **p** เป็นตัวอักษร สรุปขั้นตอนของ SAA พร้อมตัวอย่างดังตารางข้างล่างนี้

<table>
<tr><th>SAX(x, k)</th><th>ตัวอย่าง SAX([1, 1, 2, 1, 1], 2)</th></tr>
<tr><td>

- สร้างลิสต์ใหม่ชื่อ $z = [z_0, z_1, \ldots, z_{n-1}]$ เก็บ z-score ของข้อมูลในลิสต์ $x = [x_0, x_1, \ldots, x_{n-1}]$

  วิธีหา z-score เป็นดังนี้
  - หา μ ของข้อมูลทั้งหมดใน **x** โดย $\mu = \frac{1}{n}\sum_{k=0}^{n-1} x_k$
  - หา σ ของข้อมูลทั้งหมดใน **x** โดย $\sigma = \sqrt{\frac{1}{n}\sum_{k=0}^{n-1}(x_k-\mu)^2}$
  - จะได้ z-score คือ $z_k = \frac{x_k-\mu}{\sigma}$ (จะไม่มีกรณี $\sigma = 0$)

</td><td>

$x = [1, 1, 2, 1, 1]$, $k = 2$

$\mu = (1 + 1 + 2 + 1 + 1) / 5 = 1.2$

$\sigma = \sqrt{((1 - 1.2)^2+(1 - 1.2)^2+(2 - 2)^2+(1 - 1.2)^2+(1 - 1.2)^2)/5)}$
$= 0.4$

$z = [\frac{1-1.2}{0.4}, \frac{1-1.2}{0.4}, \frac{2-1.2}{0.4}, \frac{1-1.2}{0.4}, \frac{1-1.2}{0.4}]$
$= [\ -0.5,\ -0.5,\ 2,\ -0.5,\ -0.5\ ]$

</td></tr>
<tr><td>

- สร้างลิสต์ใหม่ชื่อ $p = [p_0, p_1, \ldots, p_M]$ โดยที่ **M** คือ **n/k** ปัดเศษขึ้น

  $p_j = (z_{kj} + z_{kj+1} + \ldots + z_{kj+k-1}) / k$

  คือแบ่งข้อมูลที่ติดกันใน **z** เป็นชุด ชุดละ **k** ตัว แล้วหาค่าเฉลี่ย ส่วน p_M เป็นค่าเฉลี่ยของชุดสุดท้าย (อาจมีไม่ถึง **k** ตัว)

</td><td>

$z = [\ -0.5,\ -0.5,\ 2,\ -0.5,\ -0.5\ ]$, $k = 2$

$p = [\ (-0.5-0.5)/2,\ (2-0.5)/2,\ -0.5\ ]$

$= [\ -0.5,\ 0.75\ -0.5\ ]$

</td></tr>
<tr><td>

- นำแต่ละค่าใน **p** ไปแปลงเป็นตัวอักษรโดยใช้ตารางข้างล่างนี้

ค่าของ p_j	ตัวอักษรผลลัพธ์
$-\infty < p_j \le -0.84$	a
$-0.84 < p_j \le -0.25$	b
$-0.25 < p_j \le 0.25$	c
$0.25 < p_j \le 0.84$	d
$0.84 < p_j < \infty$	e

</td><td>

$p = [\ -0.5,\ 0.75\ -0.5\ ]$
(b, d, b)

สรุป $x = [1, 1, 2, 1, 1]$, $k = 2$
ผลลัพธ์คือ **bdb**

</td></tr>
</table>

จงเขียนโปรแกรมรับรายการของจำนวนจริง (ที่แทนอนุกรมเวลา) เพื่อแปลงสตริงของตัวอักษร ด้วยวิธีข้างต้น

ข้อมูลนำเข้า

บรรทัดแรก เป็นจำนวนเต็มของค่า **k**

บรรทัดสองเป็นรายการของจำนวนจริง (คั่นด้วยช่องว่าง) ที่แทนอนุกรมเวลา

ข้อมูลส่งออก

สตริงที่แปลงมาจากอนุกรมเวลาและค่า **k** ที่ได้รับ ด้วยวิธีข้างต้น

ตัวอย่าง

Input (จากแป้นพิมพ์)	**Output** (ทางจอภาพ)
2 1 1 2 1 1	bdb
2 1.1 2.2 3.3 4.4 10 10 1 2 3 4 5 4 3 2 1	bcebcdba
3 14 13 12 9 6 5 4 5 7 10 13 17 19 19 18 16 14 13 14 15 15 16 16	caaceddd

P-47: การแยกให้เป็นลำดับเลขคณิต

ถ้าเรานำจำนวนในลำดับเลขคณิต มาเขียนติดกันหมด จะอ่านลำบาก เช่น ลำดับคือ **1, 11, 21, 31** มาเขียนติดกันหมดจะได้ **1112131** แต่ถ้าคิดสักครู่ เราก็ยังสามารถแยก **1112131** กลับมาเป็นลำดับเลขคณิตเดิมได้

จงเขียนโปรแกรมรับตัวเลขที่เขียนติดกันจำนวนหนึ่ง เพื่อหาว่า เลขชุดนี้ได้มาจากลำดับเลขคณิตใด ถือว่า ลำดับเลขคณิตที่ต้องการหา

- เป็นลำดับของจำนวนเต็มบวกหรือศูนย์
- มีจำนวนในลำดับอย่างน้อย **3** จำนวน
- มีผลต่างร่วมที่มีค่าไม่เป็นลบ (ผลต่างร่วม คือผลต่างของตัวที่ติดกัน ตัวขวาลบด้วยตัวซ้าย)

ในกรณีที่สามารถหาลำดับเลขคณิตได้หลายแบบ ให้เลือกแบบที่จำนวนแรกในลำดับมีค่าน้อยสุด เช่น **12345678** สามารถแยกเป็น **1**, **2**, **3**, **4**, **5**, **6**, **7**, **8** กับ **12**, **34**, **56**, **78** ให้ตอบแบบแรก

ข้อมูลนำเข้า

บรรทัดเดียวมีแต่ตัวเลขเขียนติดกันหมด

ข้อมูลส่งออก

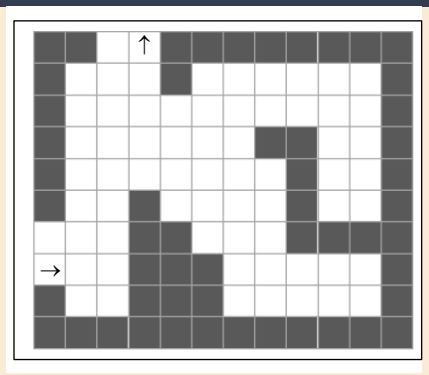
ลำดับเลขคณิตที่มีข้อกำหนดตามที่เขียนไว้ข้างบน ในกรณีที่ไม่มีลำดับคณิตที่เป็นไปได้ตามข้อกำหนด ให้แสดงคำว่า **None**

ตัวอย่าง

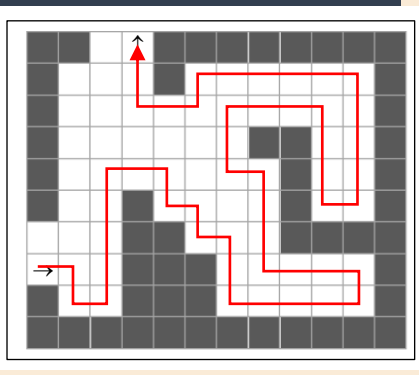
Input (จากแป้นพิมพ์)	**Output** (ทางจอภาพ)
0000000	0, 0, 0, 0, 0, 0, 0
12345678	1, 2, 3, 4, 5, 6, 7, 8
246810121416	2, 4, 6, 8, 10, 12, 14, 16
11111112221133311	11, 11111, 22211, 33311
1155555111099166643	11, 55555, 111099, 166643
999	9, 9, 9
4321	None
2020	None

P-48: ทางเดินเลียบกำแพง

ห้อง ๆ หนึ่งมีแต่กำแพง ที่ว่าง ประตูเข้า และประตูออก ดังตัวอย่างทางขวานี้ (ผังห้องถูกแบ่งเป็นตาราง ช่องเข้มคือกำแพง ช่องขาวคือที่ว่าง ประตูเข้าออกคือช่องว่างที่ขอบ)



โจทย์ข้อนี้ให้หาทางเดินเลียบกำแพงในลักษณะที่ให้มือขวาของผู้เดินแตะกำแพงตลอดเวลาจากประตูเข้าไปถึงประตูออก (โดยไม่ทับทางเดิมที่เคยผ่าน)



ขอแทนแผนผังของห้อง ด้วยสตริงหลาย ๆ ตัว แต่ละสตริงประกอบตัว **X** (แทนกำแพง) กับ **.** (แทนที่ว่าง) ดังตัวอย่างทางขวา

```
X X . . X X X X X X X X
X . . . X . . . . . . X
X . . . . . . . . . . X
X . . . . . . X X . . X
X . . . . . . . X . . X
X . . X . . . . X . . X
. . . X X . . . X X X X
. . . X X X . . . . . X
X . . X X X . . . . . X
X X X X X X X X X X X X
```

ทางเดินจากประตูเข้าไปประตูออก แทนด้วยตัวเลขตามลำดับ **0,1,2,...,9,0,1,2,...,9,0, ...** (ดังตัวอย่างทางขวา แสดงด้วยเลขสีแดง)

```
X X . 0 X X X X X X X X
X . . 9 X 5 4 3 2 1 0 X
X . . 8 7 6 9 0 1 2 9 X
X . . . . . 8 X X 3 8 X
X . 7 8 9 . 7 6 X 4 7 X
X . 6 X 0 1 . 5 X 5 6 X
. . 5 X X 2 3 4 X X X X
0 1 4 X X X 4 3 2 1 0 X
X 2 3 X X X 5 6 7 8 9 X
X X X X X X X X X X X X
```

ข้อมูลนำเข้า

- บรรทัดแรก **N** เป็นจำนวนเต็ม บอกว่า ผังห้องมีแถวแนวนอนกี่แถว
- **N** บรรทัดตามมาเป็นสตริงที่มีความยาวเท่ากันหมด ประกอบด้วยตัว **X** กับเครื่องหมายจุดเท่านั้น
- บรรทัดที่ **N+2** เป็นจำนวนเต็ม **2** จำนวน บอกเลขแถว และเลขคอลัมน์ของประตูเข้า
- บรรทัดที่ **N+3** เป็นจำนวนเต็ม **2** จำนวน บอกเลขแถว และเลขคอลัมน์ของประตูออก
- หมายเหตุ:
  - เลขแถวบนสุด และเลขคอลัมน์ซ้ายสุด คือ **0**
  - ให้ถือว่า ที่ว่างทุก ๆ ที่ของผังห้องในโจทย์ข้อนี้ จะมีที่ว่างอีกช่องอยู่ติดด้วยเสมอ

ข้อมูลส่งออก

- ถ้าหาทางเดินไปยังประตูออกไม่ได้ ให้แสดงคำว่า **Fail**
- ถ้าหาทางเดินได้ ให้แสดงผังห้อง ที่มีการเติมตัวเลขตามทางเดินจากประตูเข้าไปยังประตูออก ในลักษณะที่อธิบายข้างต้น

ตัวอย่าง

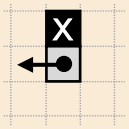
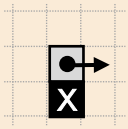
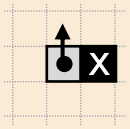
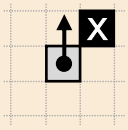
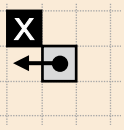
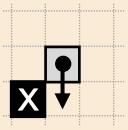
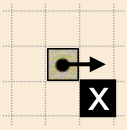
input (จากแป้นพิมพ์)	**output** (ทางจอภาพ)
4 **XXX..XX** **.....X** **.....X** **XXXXXXX** **2 0** **0 4**	**XXX.8XX** **....76X** **012345X** **XXXXXXX**
4 **X..XXXX** **X..X..X** **X..X..X** **XXXX..X** **3 5** **0 2**	**Fail**

10 xx..xxxxxxxx x...x.....x x.........x x......xx..x x.......x..x x..x....x..x ...xx...xxxx ...xxx.....x x..xxx.....x xxxxxxxxxxxx 7 0 0 3	xx.0xxxxxxx x..9x543210x x..87690129x x.....8xx38x x.789.76x47x x.6x01.5x56x ..5xx234xxxx 014xxx43210x x23xxx56789x xxxxxxxxxxxx

ข้อแนะนำ

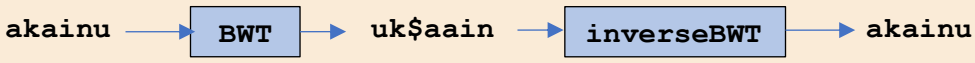
การจะตัดสินใจว่า จะไปด้านซ้าย ขวา ขึ้น หรือลง นั้น สามารถแบ่งเป็น **8** กรณี ตามรูปข้างล่างนี้ (ให้พิจารณาตามลำดับจากกรณี **1** ไป **8**) ช่องดำที่มีตัว **x** ก็คือกำแพง ช่องที่มีจุดกลมข้างใน คือช่องที่ตำแหน่งปัจจุบัน ปลายลูกศรคือ ช่องที่จะเดินไป ขออธิบายเป็นตัวอย่างดังนี้

- กรณีที่ **1** ถ้าข้างบนของช่องปัจจุบันคือกำแพง ก็ให้ไปทางซ้าย (ถ้าช่องซ้ายเป็นที่ว่าง)
- กรณีที่ **2, 3** และ **4** คล้ายกับกรณีที่ **1**
- กรณีที่ **5** ตอนนี้อยู่ที่ช่อง **r, c** แล้วช่องที่ **r-1,c+1** เป็นกำแพง (แสดงว่าก่อนนี้มาจากช่องขวา) ก็ให้เดินขึ้น (ถ้าช่องบนเป็นที่ว่าง)
- กรณีที่ **6, 7** และ **8** คล้ายกับกรณีที่ **5**
- ถ้าไม่ใช่ทุกกรณีข้างบน แสดงว่า เดินไม่ได้

(1)  กำแพงอยู่ด้านบน มีช่องว่างด้านซ้าย ก็ไปขวา	(2)  กำแพงอยู่ด้านซ้าย มีช่องว่างด้านล่าง ก็ลง	(3)  กำแพงอยู่ด้านล่าง มีช่องว่างด้านขวา ก็ไปขวา	(4)  กำแพงอยู่ด้านขวา มีช่องว่างด้านบน ก็ขึ้น
(5)  ตอนนี้อยู่ที่ช่อง `r, c` มีกำแพงที่ช่อง `r-1, c+1` ที่ช่อง `r-1, c` ว่าง ก็ขึ้น	(6) 	(7) 	(8) 

P-49: การแปลงข้อความด้วยวิธี Burrow-Wheeler Transformation

นายมังกี้ ดี ลูฟี่ ต้องการส่งข้อความไปให้เพื่อนพ้องโจรสลัดหมวกฟางของเขา แต่กลัวโดนหน่วยข่าวกรองของรัฐบาลโลกจับได้ มิเช่นนั้นจะอดเป็นราชาของโจรสลัด พวกเขาจึงตกลงวิธีที่แปลงข้อความที่จะส่ง (ด้วยวิธีที่มีชื่อว่า **BWT**) และวิธีแปลงข้อความกลับเพื่อจะอ่าน (ด้วยวิธีที่ชื่อ **inverseBWT**) จะได้ไม่โดนจับได้ เช่น ถ้าจะส่งข้อความ **akainu** ให้เพื่อน ก็ใช้วิธี **BWT** แปลงข้อความเป็น **uk$aain** แล้วส่งให้เพื่อน พอเพื่อนได้รับ ก็ใช้ **inverseBWT** แปลงกลับได้ข้อความเดิม (ดูรูปข้างล่าง)



จงเขียนฟังก์ชัน **BWT(x)** และ **inverseBWT(z)** ในโครงของโปรแกรมข้างล่างนี้

```
def BWT(x):
    ???

def inverseBWT( z ):
    ???

exec(input().strip())   # DON'T REMOVE THIS LINE
```

ฟังก์ชัน **BWT(x)** คืนสตริงที่ได้จากการแปลง **x** ด้วยวิธีข้างล่างนี้ (**x** เป็นสตริงมีแต่ตัวอักษรอังกฤษกับตัวเลข ไม่มีสัญลักษณ์พิเศษ)	ตัวอย่างเช่น **BWT("akainu")**
1. ให้ **y** คือสตริงที่ได้จากการเพิ่มเครื่องหมาย **$** ต่อท้าย **x**	**y** มีค่าเป็น **"akainu$"**
2. สร้างตารางที่เก็บสตริงต่าง ๆ ที่ได้จากการ "หมุน" **y** ไปทางซ้ายทีละ 1 ตัวอักษร ที่เป็นไปได้ทั้งหมด เช่น **akainu$** หมุนซ้าย 1 ครั้งได้ **kainu$a**	**akainu$** **kainu$a** **ainu$ak** **inu$aka** **nu$akai** **u$akain** **$akainu**
3. เรียงลำดับสตริงในตารางจากน้อยไปมากตามพจนานุกรม	**$akainu** **ainu$ak** **akainu$** **inu$aka** **kainu$a** **nu$akai** **u$akain**
4. สตริงผลลัพท์ของวิธี **BWT** ก็คือ สตริงที่สร้างจากการนำตัวอักษรตัวสุดท้ายของแต่ละสตริงในตารางมาต่อกัน	**$akainu** **ainu$ak** **akainu$** **inu$aka** **kainu$a** **nu$akai** **u$akain** **uk$aain**

ฟังก์ชัน **inverseBWT(z)** คืนสตริงที่ได้จากการแปลง **z** ด้วยวิธีข้างล่างนี้	ตัวอย่างเช่น **inverseBWT("uk$aain")**
1. ให้ **y** เป็นลิสต์ที่แต่ละช่องเป็นสตริงว่าง มีจำนวนช่องเท่ากับความยาวของสตริง **z**	**y = ['', '', '', '', '', '', '']**
2. นำแต่ละตัวใน **z** มาต่อด้านหน้าของแต่ละตัวใน **y**	**y = ['u', 'k', '$', 'a', 'a', 'i', 'n']**
3. เรียงลำดับ **y** จากน้อยไปมากตามพจนานุกรม	**y = ['$', 'a', 'a', 'i', 'k', 'n', 'u']**
4. ทำขั้นตอนที่ **2** และ **3** ซ้ำ ๆ จนสตริงแต่ละช่องใน **y** ยาวเท่ากับ **z**	ดูตัวอย่างข้างล่าง
5. หลังจากทำเสร็จ ผลลัพธ์คือสตริงในช่องแรกของ **y** (ที่ไม่เอาตัวแรกในสตริง)	

การเปลี่ยนแปลงของค่าในลิสต์ **y** เป็นดังแสดงข้างล่างนี้ (ขอแสดงค่าในลิสต์ตามแนวตั้ง)

2 และ **3** ข้างล่างนี้คือผลของ **y** หลังทำขั้นตอนที่ **2** และ **3** ในแต่ละรอบ

2	3	2	3	2	3	2	3	2	3	2	3	2	3
u	$	u$	$a	u$a	$ak	u$ak	$aka	u$aka	$akai	u$akai	$akain	u$akain	$akainu
k	a	ka	ai	kai	ain	kain	ainu	kainu	ainu$	kainu$	ainu$a	kainu$a	ainu$ak
$	a	$a	ak	$ak	aka	$aka	akai	$akai	akain	$akain	akainu	$akainu	akainu$
a	i	ai	in	ain	inu	ainu	inu$	ainu$	inu$a	ainu$a	inu$ak	ainu$ak	inu$aka
a	k	ak	ka	aka	kai	akai	kain	akain	kainu	akainu	kainu$	akainu$	kainu$a
i	n	in	nu	inu	nu$	inu$	nu$a	inu$a	nu$ak	inu$ak	nu$aka	inu$aka	nu$akai
n	u	nu	u$	nu$	u$a	nu$a	u$ak	nu$ak	u$aka	nu$aka	u$akai	nu$akai	u$akain

ข้อมูลนำเข้า

คำสั่ง **Python** หนึ่งบรรทัดที่แสดงผลจากการเรียกฟังก์ชัน **BWT** และ/หรือ **inverseBWT**

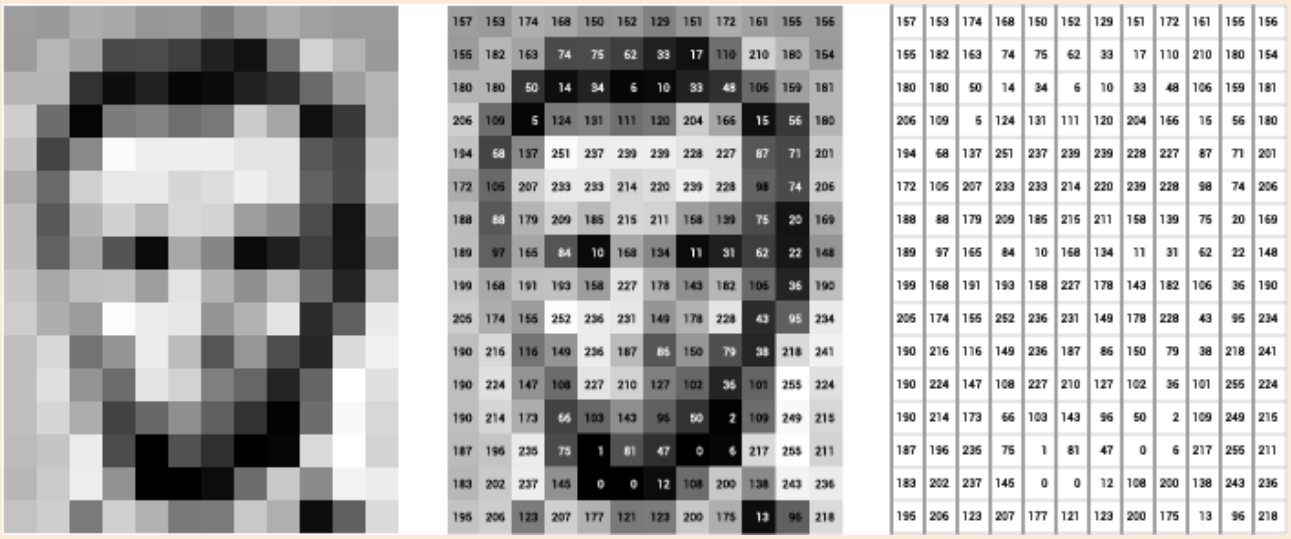
ข้อมูลส่งออก

ผลลัพธ์ที่ได้จากการทำคำสั่งที่ให้เป็นอินพุต

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
`print(BWT("akainu"), BWT("papaya"))`	`uk$aain aypp$aa`
`print(BWT("nanananananananananan"))`	`nnnnnnnnnnaaaaaaaaa$`
`print(BWT("nnnnnnnnnaaaaaaaaaa"))`	`aaaaaaaaaannnnnnnnn$`
`print(inverseBWT("uk$aain"), inverseBWT("aypp$aa"))`	`akainu papaya`
`print(inverseBWT("nnnnnnnnnnaaaaaaaaa$"))`	`nanananananananananan`
`print(inverseBWT("aaaaaaaaaannnnnnnnn$"))`	`nnnnnnnnnaaaaaaaaaa`
`print(inverseBWT(BWT('bananainpajamas')))`	`bananainpajamas`

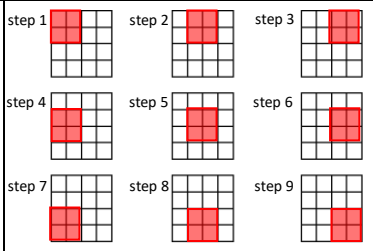
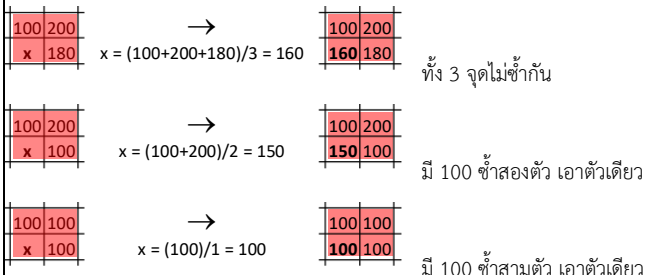
P-50: การซ่อมแซมภาพ



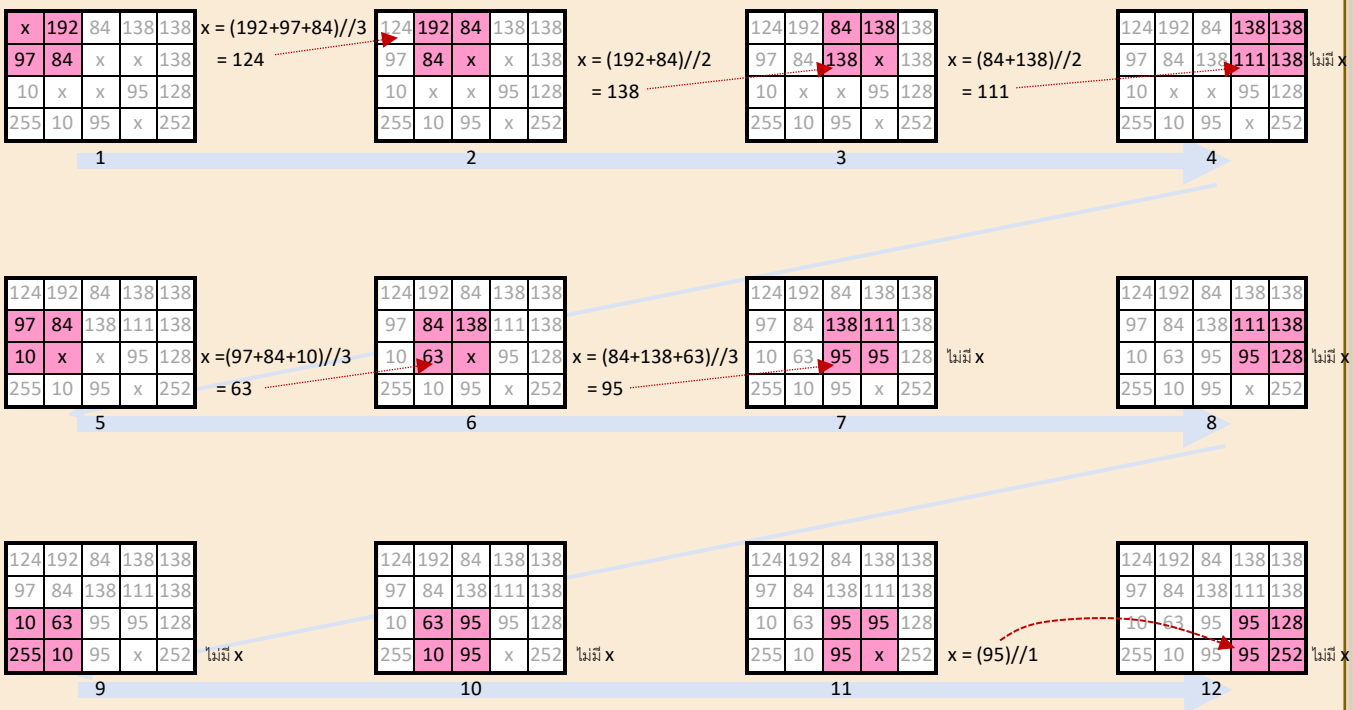
ภาพสเกลสีเทา (Grayscale image) ในระบบดิจิตอลถูกจัดเก็บในตารางที่แบ่งเป็นช่องเล็ก ๆ ตามความละเอียดของภาพ หนึ่งช่องเก็บหนึ่ง*จุดภาพ* หรือที่เรียกว่า *พิกเซล* (pixel) ความเข้มสีของแต่ละพิกเซลมีค่าระหว่าง 0 ถึง 255 โดย 0 แทนสีดำสนิท และ 255 แทนสีขาว ค่าที่อยู่ระหว่าง 0 และ 255 คือระดับความเข้มของสีเทา

ดอกเตอร์ฟุยฟุยเป็นนักวิทยาศาสตร์ที่ชอบถ่ายภาพสเกลสีเทาเป็นงานอดิเรก แต่กล้องไม่ดี ทำให้รูปภาพของเธอเกิดความเสียหายบางส่วน เธอรู้วิธีที่จะซ่อมแซมภาพ แต่เขียนโค้ดไม่เป็น!!! เธอจึงวานให้นิสิตช่วยเขียนโค้ดตามขั้นตอนสุดพิลึกของเธอเพื่อซ่อมแซมภาพส่วนที่หายไป

กำหนดให้ภาพมีขนาด N x M พิกเซล พิกเซลปกติทั่วไปมีค่า 0 ถึง 255 ถ้าพิกเซลใดที่เสียหาย จะให้มีค่าพิเศษขอเขียนว่า x
ขั้นตอนการหาค่าให้กับพิกเซลที่เสียของ ดร. ฟุยฟุย เป็นดังนี้

ขั้นตอน	ตัวอย่าง
• พิจารณาพิกเซลของภาพทีละ 4 จุดที่ติดกันในบริเวณขนาด 2x2 เริ่มจากมุมซ้ายบน ไล่จากซ้ายไปขวา ทำทีละแถว จากบนลงล่าง จนครบ (ดูรูปทางขวา)	 ภาพในตัวอย่างนี้ มีขนาด 4x4 จะมีการพิจาณากลุ่มพิกเซล 2x2 เป็นจำนวน (4-1)(4-1) = 9 กลุ่ม
• แต่ละครั้งที่พิจารณากลุ่มพิกเซล 2x2 ถ้าไม่มีจุด x ก็ไม่ต้องทำอะไร แต่ถ้ามีจุด x ให้แทนค่า x นี้ด้วย**ค่าเฉลี่ยของค่าที่แตกต่างกันของ 3 จุดที่เหลือในสี่เหลี่ยม 2x2** (ให้ถือว่า ขณะประมวลผลสี่เหลี่ยม 2x2 ใด จะมีจุด x อย่างมากแค่หนึ่งจุดในสี่เหลี่ยม 2x2) หมายเหตุ: ให้ปัดเศษหลังจุดทศนิยมของค่าเฉลี่ยทิ้งเลย เพื่อให้ค่าที่ได้เป็นจำนวนเต็ม	

ดูตัวอย่างการเปลี่ยนแปลงค่าในตารางพิกเซลของภาพตัวอย่างในหน้าถัดไป



ข้อมูลนำเข้า

บรรทัดแรกมีสองจำนวน **N** กับ **M** ระบุจำนวนแถวและจำนวนคอลัมน์ของภาพขนาด **N** × **M**

N บรรทัดต่อมา แต่ละบรรทัดคือค่าของความเข้มสีของพิกเซลของแต่ละแถวของภาพ (คั่นด้วยช่องว่าง)

สำหรับพิกเซลที่เสีย จะมีค่าเป็นตัวอักษร **x**

ข้อมูลส่งออก

ค่าของแต่ละพิกเซลหลังจากคำนวณค่าให้กับพิกเซลที่มีค่า **x** ของภาพที่รับมา แสดงค่าเป็นแถว ๆ (ดูตัวอย่างประกอบ)

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
2 2 127 127 97 x	127 127 97 112
3 3 255 0 x 0 x 127 255 x x	255 0 63 0 127 127 255 127 127
4 4 6 x 47 15 110 25 x x x 36 30 x x 36 32 x	6 47 47 15 110 25 36 32 57 36 30 32 46 36 32 31
5 5 x 192 84 138 138 97 84 x x 138 10 x x 95 128 255 10 95 x 252 121 10 185 192 255	124 192 84 138 138 97 84 138 111 138 10 63 95 95 128 255 10 95 95 252 121 10 185 192 255

P-51: คำผวน

วิธีผวนคำในวลีที่ง่ายสุด ๆ วิธีหนึ่ง ทำได้โดย ทำเฉพาะกับคำแรกกับคำสุดท้าย (ในที่นี้สนใจเฉพาะคำแรกและคำสุดท้ายที่มีหนึ่งพยางค์เท่านั้น) โดยสลับพยัญชนะต้นของสองคำนี้ แล้วก็สลับตำแหน่งของคำทั้งสอง เช่น *รักนะคนดี* ก็สลับพยัญชนะต้นของ *รัก* กับ *ดี* เป็น *ดั๊ก* กับ *รี* (ต้องปรับวรรณยุกต์ตามไปด้วย) แล้วก็สลับสองคำนี้ในวลีเดิม เป็น *รีนะคนดั๊ก*

โจทย์ข้อนี้ไม่ได้ยุ่งกับตัวอักษรไทย แต่ใช้คำอังกฤษออกเสียงไทยแทน กำหนดให้อินพุตเป็นตัวอักษรภาษาอังกฤษ แยกเป็นคำ ๆ (คั่นด้วยช่องว่าง) เช่น rak na khon dee ผวนได้เป็น ree na khon dak โดยให้ถือว่า พยัญชนะต้นของคำ คือลำดับตัวอักษรของคำตั้งแต่ตัวซ้ายสุดไปจนถึงตัวก่อนตัวสระ (สระคือ a, e, i, o และ u) เช่น พยัญชนะต้นของ chu คือ ch และของ khi คือ kh ดังนั้น chu wit khi ผวนเป็น chi wit khu

ข้อมูลนำเข้า

บรรทัดเดียวเป็นสตริงของวลีที่ประกอบด้วยคำต่าง ๆ คั่นด้วยช่องว่าง (สตริงนี้มีแต่ตัวอักษรอังกฤษตัวเล็ก กับช่องว่างเท่านั้น)

ข้อมูลส่งออก

คำผวนของวลีที่ได้รับ ตามวิธีที่นำเสนอข้างต้น

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
`mee ther khon deaw`	`meaw ther knon dee`
`mow mai khub`	`mub mai khow`
`lun took wove`	`love took wun`
`kho hai tam dai na ja`	`kha hai tam dai na jo`

P-52: คำที่หมุนให้เป็นพาลินโดรมได้

พาลินโดรม (palindrome) คือลำดับอักษรที่เมื่ออ่านจากซ้ายไปขวาเหมือนกับอ่านจากขวามาซ้าย (ถือว่าตัวอักษรภาษาตัวพิมพ์เล็กเหมือนตัวพิมพ์ใหญ่) เช่น **`Civic, Noon, RaceCar`**

ให้ **`t`** เป็นสตริง ปัญหาที่น่าสนใจคือ เราจะ "หมุน" ลำดับตัวอักษรใน **`t`** จนเป็นพาลินโดรมได้หรือไม่ การหมุนคือ การเลื่อนตัวอักษรทุกตัวไปทางซ้ายหนึ่งตำแหน่งโดยตัวซ้ายสุดจะย้ายมาอยู่ทางขวาสุด เช่น **`ICCIV`** ไม่ใช่พาลินโดรม ถ้าเราหมุนไปทางซ้ายหนึ่งครั้งจาก **`ICCIV`** จะได้ **`CCIVI`** ก็ยังไม่ใช่พาลินโดรม แต่ถ้าหมุนอีกครั้ง จะได้ **`CIVIC`** ซึ่งเป็นพาลินโดรม แสดงว่า **`ICCIV`** หมุนจนเป็นพาลินโดรมได้ แต่สำหรับสตริง **`"ABC"`** หมุนยังไงก็ไม่ได้พาลินโดรม

จงเขียนโปรแกรมรับสตริงแล้วตรวจว่า สามารถหมุนเป็นพาลินโดรมได้หรือไม่

ข้อมูลนำเข้า

ข้อความที่ประกอบด้วยตัวอักษรอังกฤษตัวพิมพ์ใหญ่หรือเล็กเท่านั้น

ข้อมูลส่งออก

แสดง **`Y`** ถ้าสตริงที่รับมาสามารถหมุนจนเป็นพาลินโดรมได้

แสดง **`N`** ถ้าสตริงที่รับมาไม่สามารถหมุนจนเป็นพาลินโดรมได้

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
`A`	**`Y`**
`Abba`	**`Y`**
`aAaaB`	**`Y`**
`HaHaHa`	**`N`**

P-53: ข้อความที่มีครบทุกตัวอักษร (Pangram)

ข้อความหนึ่งเป็น pangram ก็เมื่อข้อความนั้นมีตัวอักษรอังกฤษ (a ถึง z ใหญ่หรือเล็กก็ได้) ปรากฏครบทุกตัว (อย่างน้อย 1 ตัว) เช่น

- **`The quick brown fox jumps over the lazy dog.`**
- **`Pack my box with five dozen liquor jugs.`**

จงเขียนโปรแกรมรับข้อความ 1 บรรทัด แล้วตรวจว่า ข้อความนี้เป็น pangram หรือไม่

ข้อมูลนำเข้า

ข้อความที่ประกอบด้วยตัวอักษรภาษาอังกฤษ และอาจมีตัวอักขระอื่นที่ไม่ใช่ตัวอักษรอังกฤษ (ดูตัวอย่างข้างล่าง)

ข้อมูลส่งออก

แสดงตัวอักษร **`Y`** ถ้าสตริงที่รับมาเป็น pangram ถ้าไม่ใช่ แสดงตัวอักษร **`N`**

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
`The quick brown fox jumps over the lazy dog !!!`	`Y`
`Pack my box with five dozen liquor jugs.`	`Y`
`Abcdefghijklmnopqrstuvwxyz is definitely a pangram.`	`Y`
`Am I am a pangram ?`	`N`

P-54: ข้อความที่ใช้ตัวอักษรต่างกันหมด (Heterogram)

ข้อความหนึ่งเป็น heterogram ก็เมื่อข้อความนั้นมีตัวอักษรอังกฤษไม่ซ้ำกันเลย (ถือว่าตัวอักษรอังกฤษตัวพิมพ์ใหญ่กับตัวพิมพ์เล็กเหมือนกัน) เช่น

- **`Dermatoglyphics`**
- **`The big dwarf only jumps.`**

แต่ **`PythoN`** ไม่ใช่ heterogram

จงเขียนโปรแกรมรับข้อความ 1 บรรทัด แล้วตรวจว่า ข้อความนี้เป็น heterogram หรือไม่

ข้อมูลนำเข้า

ข้อความที่ประกอบด้วยตัวอักษรภาษาอังกฤษ และอาจมีตัวอักขระอื่นที่ไม่ใช่ตัวอักษรอังกฤษก็ได้ (ดูตัวอย่างข้างล่าง)

ข้อมูลส่งออก

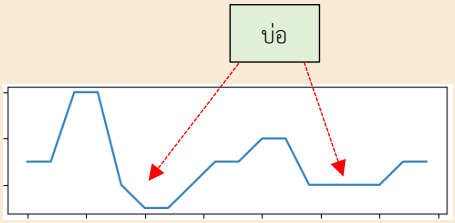
แสดง **`Y`** ถ้าสตริงที่รับมาเป็น heterogram ถ้าไม่ใช่ แสดง **`N`**

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
`The big dwarf only jumps.`	**`Y`**
`--Dermatoglyphics--`	**`Y`**
`uncopyrightables`	**`Y`**
`JavA`	**`N`**

P-55: จำนวนบ่อ

นิยาม "บ่อ" ในลิสต์ คือ บริเวณในลิสต์ที่ข้อมูลจากซ้ายไปขวาเริ่มด้วยค่าที่ลดลง จากนั้นอาจจะมีเท่ากันบ้าง ลดลงบ้าง แล้วก็มีค่าเพิ่มขึ้น เช่น ลิสต์ [5, 5, 8, 8, 4, 3, 3, 4, 5, 5, 6, 6, 4, 4, 4, 4, 5, 5] มี 2 บ่อ ดูรูปข้างล่างนี้ที่ได้จากการนำค่าในลิสต์ไปวาดกราฟประกอบ (แกน x เป็นอินเด็กซ์ แกน y คือข้อมูลที่อินเด็กซ์นั้น ๆ ของลิสต์)



จงเขียนโปรแกรมที่รับรายการของจำนวน แล้วนับจำนวนบ่อในลิสต์นี้

ข้อมูลนำเข้า

รายการของจำนวน คั่นด้วยช่องว่าง

ข้อมูลส่งออก

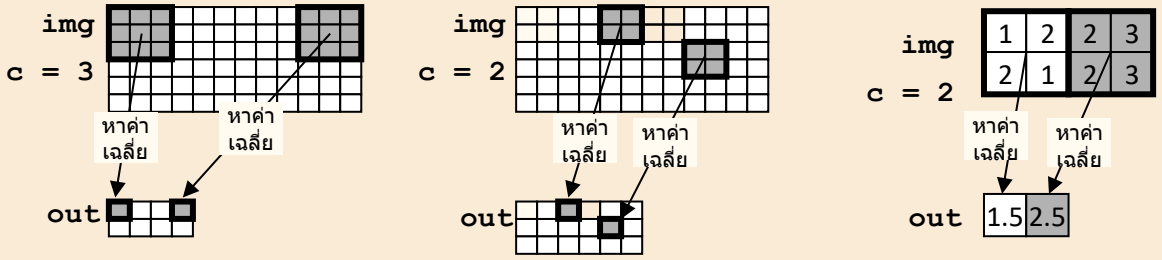
จำนวนบ่อที่พบในรายการที่รับเข้ามา

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
1 2 3 4 5 6 7	0
4 4 4 4 5 5 5 4 4 4	0
5 5 5 4 4 4 4 5 5 5	1
9 1 9 1 9 1 9 1 9 1	4
5 5 8 8 4 3 3 4 5 5 6 6 4 4 4 4 5 5	2

P-56: การปรับขนาดภาพ

ฟังก์ชัน **scale(img, c)** รับรูปภาพ **img** ที่เป็น numpy อาเรย์ 2 มิติ เพื่อคืน numpy อาเรย์ 2 มิติ (ให้ชื่อว่า **out**) ที่มีจำนวนแถวและจำนวนหลักลดลง **c** เท่าของ **img** (เช่น **img** มีขนาด 10×15 และ **c = 5** จะได้ **out** ที่มีขนาด 2×3) ให้ถือว่า **c** > 0 และจำนวนแถวและจำนวนหลักของ **img** หารด้วย **c** ได้ลงตัว สำหรับค่าในแต่ละช่องของ **out** นั้นหาได้จากค่าเฉลี่ยของข้อมูลในอาเรย์ย่อยขนาด **c** × **c** ของ **img** ในลักษณะที่แสดงเป็นตัวอย่างข้างล่างนี้




```
import numpy as np

def scale(img, c) :

  ???              # เขียนตรงนี้

def read_img():
  row, col = [int(e) for e in input().split()]
  img = np.ndarray((row,col))
  for i in range(row):
    img[i] = [float(e) for e in input().split()]
  return img

def show_output(out):
  for i in range(out.shape[0]):
    print(" ".join([str(e) for e in out[i]]))

img = read_img()
c = int(input())
out = scale(img, c)
show_output(out)
```

ข้อมูลนำเข้า

บรรทัดแรกเป็นจำนวนเต็ม 2 จำนวน, Row กับ Col บอกจำนวนแถวกับจำนวนคอลัมน์ ตามลำดับ

Row บรรทัดต่อมา หนึ่งบรรทัดแทนข้อมูลในเมทริกซ์หนึ่งแถว ประกอบด้วยจำนวนจริง Col จำนวน

บรรทัดสุดท้ายเป็นจำนวนเต็ม 1 จำนวน แทนค่า c ที่อธิบายข้างต้น

ข้อมูลส่งออก

แสดงเมทริกซ์ผลลัพธ์ของฟังก์ชัน scale จำนวน Row / c บรรทัด

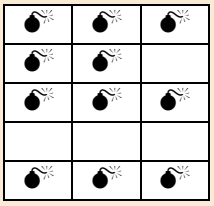
แต่ละบรรทัดแสดงจำนวนจริง Col / c จำนวน แทนค่าในเมทริกซ์ผลลัพธ์

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
1 3 3 2 2 1	3.0 2.0 2.0
3 3 3 2 2 2 0 2 2 2 3 3	2.0
2 4 1 2 2 3 2 1 2 3 2	1.5 2.5
4 6 1 2 2 3 3 2 2 1 2 3 2 3 2 3 2 2 4 2 3 2 2 2 2 4 2	1.5 2.5 2.5 2.5 2.0 3.0

P-57: จำนวนกับระเบิด (Minesweeper)

ให้เขียนโปรแกรมนับจำนวนระเบิดที่อยู่รอบ ๆ แต่ละช่อง และแสดงผล (รอบ ๆ คือทั้ง 8 ทิศ)



⇨

input

1	1	1
1	1	0
1	1	1
0	0	0
1	1	1

⇨

output

9	9	9
9	9	5
9	9	9
4	6	4
9	9	9

ข้อมูลนำเข้า

บรรทัดแรกเป็นเลขบอกขนาดตาราง (จำนวนแถว ตามด้วยจำนวนคอลัมน์ คั่นด้วยจุลภาคและช่องว่าง)

บรรทัดต่อ ๆ มา เป็นเลข 1 หรือ 0 แทนข้อมูลตาราง (คั่นด้วยช่องว่าง) 1 คือมีระเบิด 0 คือไม่มีระเบิด

ข้อมูลส่งออก

ตัวเลขมีขนาดเท่าตารางที่รับมา แต่ละช่องแสดงจำนวนระเบิดรอบ ๆ (รวมทั้งแปดทิศ) ช่องที่มีระเบิดให้แสดง 9

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
2, 2 0 0 1 0	1 1 9 1
1, 3 0 0 1	0 1 9
5, 3 1 1 1 1 1 0 1 1 1 0 0 0 1 1 1	9 9 9 9 9 5 9 9 9 4 6 4 9 9 9

P-58: รหัส Soundex

Soundex คือวิธีให้รหัสที่แทนการออกเสียงของชื่อภาษาอังกฤษ เช่น **OBAMA** มีรหัสเป็น **O150**, **TRUMP** มีรหัสเป็น **T651** เป็นต้น วิธีให้รหัสมีหลักการง่าย ๆ คือ ตัวอักษรตัวแรกคงไว้เหมือนเดิม ตัวที่เหลือให้แปลงเป็นตัวเลข ตามตารางทางขวานี้ เช่น **TRUMP** มีรหัส **T651**

ตัวอักษร	รหัสตัวเลข
B, F, P, V	1
C, G, J, K, Q, S, X, Z	2
D, T	3
L	4
M, N	5
R	6
A, E, I, O, U, H, W, Y	ไม่มีรหัส

T	R	U	M	P
T	6		5	1

แต่ก็มีเรื่องจุกจิกที่ต้องพิจารณาดังนี้

1. ถ้ามีรหัสตัวเลขเกิน **3** ตัว ให้ตัดตัวเลขทางขวาออกให้เหลือ **3** ตัว เช่น **WILKINSON** ได้ **W42525** ตัดเหลือ **W425**
2. ถ้ามีรหัสตัวเลขไม่ถึง **3** ตัว ให้เติม **0** ทางขวาจนครบ เช่น **FORD** ได้ **F63** เติม **0** เป็น **F630**
3. ถ้ามีตัว **H** กับ **W** ที่ใด (ยกเว้นตำแหน่งแรกสุด) ให้ถือว่า ไม่มี **H** กับ **W** ที่ตำแหน่งนั้น เช่น **SHAWMAN** ได้ **S550**
4. ตัวอักษรที่ติดกันในชื่อที่มีรหัสเดียวกัน ให้ถือว่า มีแค่รหัสเดียว เช่น
   - **JACKSON** ได้ **J250** เพราะ **CKS** มีรหัส **2** เหมือนกัน แทนด้วย **2** ตัวเดียว
   - **TYMCZAK** ได้ **T522** เพราะ **CZ** มีรหัส **2** เหมือนกัน แต่ **K** ไม่ติดกับ **Z** เพราะมีตัว **A** คั่นจึงเป็นรหัส **2** อีกตัว
   - **PFISTER** ได้ **P236** เพราะ **F** มีรหัสเป็น **1** เหมือนกับของ **P** (ถึงแม้ **P** ไม่ต้องเข้ารหัส แต่ก็ถือว่ามีรหัสเดียวกับ **F**) จึงไม่มีรหัสของ **F**
   - **ASHCRAFT** ได้ **A261** เพราะเสมือนไม่มี **H** ทำให้ **S** กับ **C** เสมือนอยู่ติดกัน ซึ่งทั้งสองตัวนี้อยู่กลุ่มเดียวกัน จึงแทนด้วย **2** ตัวเดียว ได้ **A2613** ตัดเหลือ **A261**

จงเขียนโปรแกรมรับชื่อ เพื่อแสดงรหัส **soundex**

ข้อมูลนำเข้า

ชื่อประกอบด้วยตัวอักษรอังกฤษตัวใหญ่หมดทุกตัว

ข้อมูลส่งออก

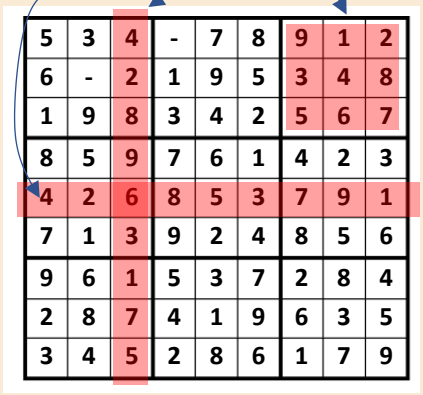
รหัส **soundex** ของชื่อที่รับ

ตัวอย่าง

Input (จากแป้นพิมพ์)	Output (ทางจอภาพ)
HENSON	H525
OBAMA	O150
SCHWARZENEGGER	S625
DIJKSTRA	D236
SCHMIDT	S530
WASHINGTON	W252
HUGHES	H220
BHYFPVCGJKQSXZDTMNLR	B123

P-59: การเติมค่าที่หายไปในกระดาน Sudoku แบบง่ายสุด ๆ

Sudoku เป็นเกมเติมตัวเลข 1 ถึง 9 ในช่องที่เว้นว่างของตารางขนาด 9x9 โดยไม่ให้เลขซ้ำกัน ใน**แนวนอน แนวตั้ง** และ**ตารางย่อย** จากตัวอย่างตารางทางขวานี้ ก็ต้องเติมช่องว่างแถวบนสุดด้วยเลข 6 และ ช่องว่างอีกช่องในแถวที่ 2 ด้วยเลข 7 ก็เป็นอันแก้ปริศนา Sudoku นี้ได้



5	3	4	-	7	8	9	1	2
6	-	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

จงเขียนโปรแกรมที่อ่านตารางขนาด 9x9 แล้วเติมตัวเลขในช่องว่างที่ยังไม่เติม ให้ถูกต้อง โดยแต่ละแถว**แนวตั้งมีช่องว่างที่ยังไม่เติมอย่างมากหนึ่งช่อง**เท่านั้น และตัวเลขในตารางเป็นตัวเลขที่ถูกต้องแล้ว

ข้อมูลนำเข้า

มี 9 บรรทัด แต่ละบรรทัดประกอบด้วยเลข **1** ถึง **9** หรือสัญลักษณ์ **-** เท่านั้น เรียงติดกันแทนเลขในแต่ละแถวแนวนอนของตาราง โดยสัญลักษณ์ **-** แทนช่องที่ยังไม่เติมตัวเลข

ข้อมูลส่งออก

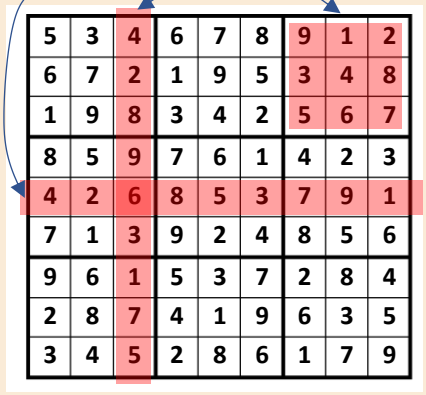
มี 9 บรรทัด แต่ละบรรทัดประกอบด้วยเลข **1** ถึง **9** เรียงติดกันแทนเลขในแต่ละแถวแนวนอนของตารางที่ถูกต้องตามกฎ

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
----26347 725341698 346978215 981257463 564139872 237684159 473815926 819762534 65249----	198526347 725341698 346978215 981257463 564139872 237684159 473815926 819762534 652493781
534-78912 67-195348 1-8342567 -59761423 4268-3791 71392485- 961537-84 287419635 34528-179	534678912 672195348 198342567 859761423 426853791 713924856 961537284 287419635 345286179

P-60: การตรวจความถูกต้องของกระดาน Sudoku

Sudoku เป็นเกมเติมตัวเลข 1 ถึง 9 ในช่องที่เว้นว่างของตารางขนาด 9x9 โดยไม่ให้เลขซ้ำกัน ใน**แนวนอน แนวตั้ง** และ**ตารางย่อย** จงเขียนโปรแกรมที่อ่านตารางที่ได้เติมตัวเลขครบแล้ว เพื่อตรวจว่าถูกตามกฎหรือไม่



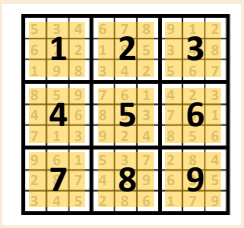
5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

ข้อมูลนำเข้า

มีทั้งหมด 10 บรรทัด (ดูตัวอย่างในหน้าถัดไปประกอบ) แต่ละบรรทัดใน 9 บรรทัดแรกประกอบด้วยเลข 9 ตัวติดกันแทนเลขในแต่ละแถวแนวนอนของตาราง โดยที่

- อาจมีแถวแนวนอนที่มีเลขผิด (ถ้ามีเลขผิดในแถวใด จะผิดแค่ตำแหน่งเดียวในแถวนั้น)
- อาจมีแถวแนวตั้งที่มีเลขผิด (ถ้ามีเลขผิดในแถวใด จะผิดแค่ตำแหน่งเดียวในแถวนั้น)
- อาจมีตารางย่อย 3x3 ที่มีเลขผิด
- ข้อสังเกต: ผิดในแถวแนวนอน แนวตั้ง หรือในตารางย่อย คือ ผิดเพราะมีเลขซ้ำกัน หรือจะคิดว่า ผิดเพราะมีเลขบางตัวหายไป ก็ได้

สำหรับบรรทัดที่ 10 (บรรทัดสุดท้ายของอินพุต) เป็นตัวอักษร 1 ตัว



1	2	3
4	5	6
7	8	9

- ถ้าเป็น **R** คือ ให้ตรวจเฉพาะแถวแนวนอน
- ถ้าเป็น **C** คือ ให้ตรวจเฉพาะแถวแนวตั้ง
- ถ้าเป็น **B** คือ ให้ตรวจเฉพาะตารางย่อย 3x3
- ถ้าเป็นตัวอื่น คือ ให้ตรวจทั้งแถวแนวนอน แนวตั้ง และตารางย่อย 3x3

ข้อมูลส่งออก

ผลลัพธ์ขึ้นกับตัวอักษรในบรรทัดที่ 10 ของอินพุต ดังนี้ (ดูตัวอย่างผลลัพธ์ในหน้าถัดไปประกอบด้วย)

- ถ้าเป็น **R** จะแสดงคำว่า **Row:** ตามด้วยหมายเลขแถวแนวนอนที่มีเลขผิดกฎ เรียงเลขแถวจากน้อยไปมาก คั่นด้วยช่องว่าง
- ถ้าเป็น **C** จะแสดงคำว่า **Col:** ตามด้วยหมายเลขแถวแนวตั้งที่มีเลขผิดกฎ เรียงเลขแถวจากน้อยไปมาก คั่นด้วยช่องว่าง
- ถ้าเป็น **B** จะแสดงคำว่า **Box:** ตามด้วยหมายเลขตารางย่อยที่มีเลขผิดกฎ เรียงเลขตารางย่อยจากน้อยไปมาก คั่นด้วยช่องว่าง
- ถ้าเป็นตัวอักษรอื่น ให้แสดง 3 บรรทัด
  - บรรทัดที่ 1 แสดงคำว่า **Row:** ตามด้วยหมายเลขแถวแนวนอนที่มีเลขผิดกฎ เรียงจากเลขแถวจากน้อยไปมาก
  - บรรทัดที่ 2 แสดงคำว่า **Col:** ตามด้วยหมายเลขแถวแนวตั้งที่มีเลขผิดกฎ เรียงจากเลขแถวจากน้อยไปมาก
  - บรรทัดที่ 3 แสดงคำว่า **Box:** ตามด้วยหมายเลขตารางย่อยที่มีเลขผิดกฎ เรียงจากเลขตารางย่อยจากน้อยไปมาก

กรณีไม่มีที่ผิด ให้แสดงคำว่า **OK** แทนที่จะแสดงหมายเลขที่ผิด

หมายเหตุ: หมายเลขแถวแนวนอนเป็น 1 ถึง 9 เรียงจากบนลงล่าง และหมายเลขแถวแนวตั้งเป็น 1 ถึง 9 เรียงจากซ้ายไปขวา ส่วนหมายเลขตารางย่อย จะเป็น 1 ถึง 9 เรียงดังรูปด้านบนนี้

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
198526347 725341698 346978215 981257463 564139872 237684159 473815926 819762534 652493781 Q	Row: OK Col: OK Box: OK

198516347 725341698 345978215 981257463 564139872 237684259 473815926 819762534 652493789 R	Row: 1 3 6 9
198516347 725341698 345978215 981257463 564139872 237684259 473815926 819762524 652493781 C	Col: 3 5 7 8
198516347 725341698 345978215 981257463 564139872 237684259 473815926 819762534 652493789 B	Box: 1 2 6 9
123456789 234567891 345678912 456789123 567891234 678912345 789123456 891234567 912345678 C	Col: OK
123456789 234567891 345678912 456789123 567891234 678912345 789123456 891234567 912345678 X	Row: OK Col: OK Box: 1 2 3 4 5 6 7 8 9

P-61: การใส่จุลภาคและจุดทศนิยมในจำนวน

จงเขียนโปรแกรมรับจำนวน อาจมีหรือไม่มีจุดทศนิยมก็ได้ แล้วก็นำมาแสดงในรูปแบบที่มีเครื่องหมายจุลภาค (comma) คั่นทุกสามหลักทางซ้ายของจุด และมีเลขทางขวาของจุดทศนิยมจำนวน 2 หลัก (ปัดเศษด้วย ถ้าเลขตัวที่สามหลังจุดทศนิยมมีค่ามากกว่าหรือเท่ากับ 5) เช่น อินพุตเป็น **`1024`** จะแสดง **`1,024.00`** อินพุตเป็น **`1234500.45501`** จะแสดง **`1,234,500.46`**

คำเตือน: โจทย์นี้ใช้ฟังก์ชัน **`round`** ในการปัดเศษไม่ได้ เช่น **`round(1.115, 2)`** ได้ **`1.11`** แต่ในโจทย์ต้องการให้การปัดแล้วได้ **`1.12`**

ข้อมูลนำเข้า

บรรทัดเดียวเป็นจำนวน (*ไม่ติดลบ*) ที่มีลักษณะดังนี้

- มีเลขอย่างน้อยหนึ่งตัว
- มีหรือไม่มีจุดทศนิยมก็ได้
- ถ้ามีจุด ทางซ้ายหรือทางขวาอาจมีหรือไม่มีเลขก็ได้ (แต่ต้องมีข้างใดข้างหนึ่ง หรือทั้งสองข้าง)
- เลขทางซ้าย และทางขวาจุด จะมีกี่หลักก็ได้

ข้อมูลส่งออก

จำนวนที่รับมาในรูปแบบที่มีเครื่องหมายจุลภาค (comma) คั่นทุกสามหลักทางซ้ายของจุด และมีเลขทางขวาจุดทศนิยมจำนวน 2 หลัก ถ้าเลขหลักที่สามหลังจุดมีค่ามากกว่าหรือเท่ากับ 5 จะปัดเศษจากหลักที่สามหลังจุดมาให้เลขหลักที่สอง (ซึ่งอาจมีการทดไปหลักทางซ้ายถัด ๆ ไปได้)

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)	กรณี
`12`	`12.00`	ไม่มีจุดทศนิยม
`0.`	`0.00`	ไม่มีเลขหลังจุดทศนิยม
`.0`	`0.00`	ไม่มีเลขหน้าจุดทศนิยม
`0.5`	`0.50`	มีเลขหลังจุดทศนิยมแค่หลักเดียว
`0.50`	`0.50`	มีเลขหลังจุดทศนิยมสองหลัก
`5.014`	`5.01`	มีเลขหลังจุดทศนิยมมากกว่าสอง แต่ไม่ปัดขึ้น
`999.996`	`1,000.00`	มีเลขหลังจุดทศนิยมมากกว่าสอง และปัดขึ้น
`123456789012345678.12499999`	`123,456,789,012,345,678.12`	จำนวนหน้าจุดมีได้ไม่จำกัด

P-62: ตัวใดในลำดับเลขคณิตที่ไม่ถูกต้อง

ลำดับเลขคณิต (arithmetic sequence) คือ ลำดับ $x_0, x_1, x_2, \ldots$ ที่ผลต่างของตัวที่ติดกันเป็นค่าคงตัว คือ $x_{i+1} - x_i$ เท่ากันหมดสำหรับทุก i

จงเขียนโปรแกรมที่รับลำดับของจำนวนมาลำดับหนึ่ง (มีอย่างน้อย 4 ตัว) โดยลำดับนี้เป็นลำดับเลขคณิต แต่ว่า อาจมีข้อมูลหนึ่งตัวในลำดับนี้ที่มีค่าผิดไป ทำให้ผลต่างของค่านี้กับค่าที่ติดกัน ไม่เหมือนกับผลต่างของคู่ที่ติดกันคู่อื่น เช่น ลำดับ **2, 4, 9, 8, 10** มีเลข **9** เป็นค่าที่ผิด (ควรเป็น **6**) หน้าที่ของโปรแกรมนี้คือ ให้แสดงอินเด็กซ์ของลิสต์ที่เก็บค่าที่ผิด (ซึ่งมีอย่างมากตัวเดียว) หรือในกรณีไม่มีตัวผิด ให้แสดงคำว่า **No error**

ข้อมูลนำเข้า

บรรทัดแรกเป็นจำนวนเต็ม **n** ตามด้วยอีก **n** บรรทัด บรรทัดละจำนวน (**n** มีค่าอย่างน้อย **4** แน่ ๆ ในข้อมูลที่ใช้ทดสอบ)

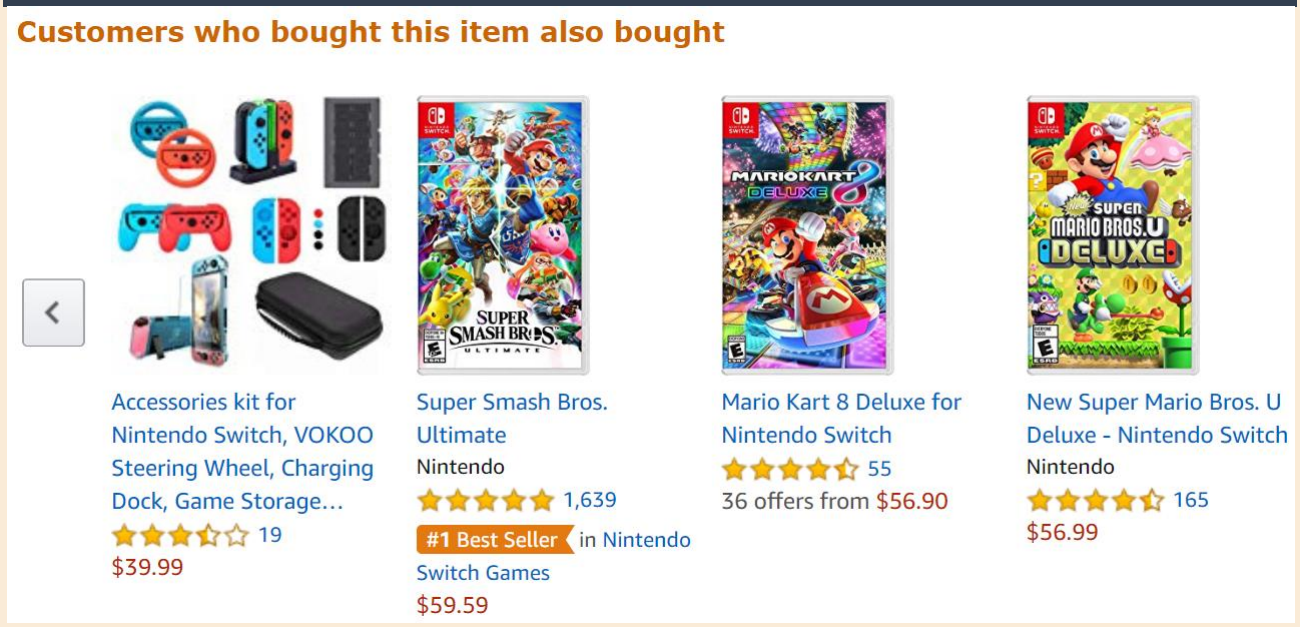
ข้อมูลส่งออก

อินเด็กซ์ของลำดับของจำนวนที่ได้รับที่เป็นข้อมูลที่ผิดไปจากลำดับเลขคณิตที่ควรเป็น (ให้ตัวแรกในลำดับคืออินเด็กซ์ **0**)
กรณีที่ไม่มีข้อมูลผิดเลย ให้แสดง **No error**

ตัวอย่าง

input (จากแป้นพิมพ์)	**output** (ทางจอภาพ)
5 10 20 30 40 50 (มีข้อมูล 5 ตัว / ข้อมูลในลำดับที่ต้องการตรวจสอบ)	**No error**
4 10 21 30 40	1
5 100 20 30 40 50	0
5 11 22 33 44 99	4

P-63: ระบบแนะนำสินค้า



เว็บไซต์ขายสินค้าทั่วไปจะแสดงรายการสินค้าที่เกี่ยวข้องกับสินค้าที่ผู้ชมดูอยู่ เช่น ในรูปข้างบนนี้แสดงอยู่ใน amazon.com แนะนำสินค้าในเพจของเครื่องเกม Nintendo Switch โดยแนะนำสินค้าที่ลูกค้าของ Amazon ที่เคยซื้อ Nintendo Switch และซื้อสินค้าเหล่านี้ด้วย

สิ่งที่ต้องทำ

เขียนโปรแกรมที่รับข้อมูลการซื้อสินค้าต่าง ๆ ของลูกค้าทั้งหลายของร้าน แล้วแสดงสินค้าที่ในอดีตมีลูกค้าซื้อไปด้วย ดังตัวอย่างข้างล่างนี้

4 A1 A2 A3 A4 A5 A4 B6 A2 A1 A9 B1 A2 A5 B2 A1 A5 A2 A4 A3 2 A1	บรรทัดแรกเป็น 4 บอกว่า มีข้อมูลรายการการซื้อสินค้าของลูกค้าในอดีต 4 คน 4 บรรทัดต่อมา แต่ละบรรทัดแสดงรายการของชื่อสินค้าที่เคยถูกซื้อโดยลูกค้าคนเดียวกัน สองบรรทัดสุดท้ายเป็น 2 กับ A1 คือ ให้หาว่า มีสินค้าใดที่มีลูกค้าตั้งแต่ 2 รายขึ้นไปในอดีต ที่ซื้อสินค้านั้น และซื้อ A1 ด้วย

วิเคราะห์จากตัวอย่างข้างบนนี้ ได้ผลดังตารางข้างล่างนี้

ชื่อสินค้า	A2	A3	A4	A5	A9	B1	B2	B6
จำนวนลูกค้าในอดีตที่เคยซื้อสินค้านี้กับ A1	3	2	3	2	0	0	0	1

ดังนั้นสินค้าที่มีลูกค้าตั้งแต่ 2 รายขึ้นไปที่เคยซื้อไปกับ A1 ก็คือ A2 A3 A4 A5

ในโจทย์นี้เราต้องการ แสดงชื่อสินค้าให้เรียงจากมากไปน้อยตามจำนวนลูกค้าที่เคยซื้อไปกับ A1
ถ้ามีค่าเท่ากันให้เรียงตามชื่อสินค้าจากน้อยไปมากตามพจนานุกรม

ดังนั้น ต้องแสดง A2 A4 A3 A5 (ในกรณีที่ไม่มีสินค้าแนะนำเลย ให้แสดงคำว่า None)

มี 3 คนที่ซื้อสินค้า A1 กับ A2
มี 3 คนที่ซื้อสินค้า A1 กับ A4

มี 2 คนที่ซื้อสินค้า A1 กับ A3
มี 2 คนที่ซื้อสินค้า A1 กับ A5

ข้อมูลนำเข้า

เป็นไปตามลักษณะที่อธิบายไว้ข้างต้น

ข้อมูลส่งออก

เป็นไปตามลักษณะที่อธิบายไว้ข้างต้น

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
4 A1 A2 A3 A4 A5 A4 B6 A2 A1 A9 B1 A2 A5 B2 A1 A5 A2 A4 A3 1 A1	A2 A4 A3 A5 B6
4 A1 A2 A3 A4 A5 A4 B6 A2 A1 A9 B1 A2 A5 B2 A1 A5 A2 A4 A3 2 A1	A2 A4 A3 A5
4 A1 A2 A3 A4 A5 A4 B6 A2 A1 A9 B1 A2 A5 B2 A1 A5 A2 A4 A3 3 A1	A2 A4
4 A1 A2 A3 A4 A5 A4 B6 A2 A1 A9 B1 A2 A5 B2 A1 A5 A2 A4 A3 4 A1	None
4 A1 A2 A3 A4 A5 A4 B6 A2 A1 A9 B1 A2 A5 B2 A1 A5 A2 A4 A3 1 A999	None

ข้อมูลที่ใช้ทดสอบจะมีค่านี้ที่มากกว่า 0 แน่ ๆ

P-64: เครือข่ายทางสังคม

เขียนโปรแกรมเพื่อวิเคราะห์ข้อมูลความสัมพันธ์ของคนใน social network

ข้อมูลนำเข้า

หลายบรรทัดประกอบด้วย ชื่อ**คนกดไลค์** คั่นด้วยเว้นวรรค ตามด้วยรายชื่อ**คนโพสต์** ที่เขาไลค์อาจมีมากกว่า 1 คน ชื่อแต่ละคนไม่ซ้ำกัน คั่นด้วยเว้นวรรค เช่น `A B H D` แปลว่า `A` กดไลค์ `B H` และ `D` (ชื่อคนกดไลค์ในแต่ละบรรทัดไม่ซ้ำกัน)

เมื่อใส่ข้อมูลครบแล้วบรรทัดถัดไปจะเป็น **done** หลังจากนั้นเป็นบรรทัดคำสั่ง 1 บรรทัด ที่มีความต้องการดังนี้

1) `R` (report all likers) แสดงรายชื่อ**คนกดไลค์** ตามลำดับตัวอักษร พร้อมจำนวนคนโพสต์ที่เขาไลค์
2) `T` (find top poster) หาว่า **คนโพสต์** คนไหน มีคนกดไลค์มากที่สุด ถ้ามีมากกว่า 1 คน ให้แสดงรายชื่อ **คนโพสต์** เหล่านี้ตามลำดับตัวอักษร
3) `C` **คนโพสต์1 คนโพสต์2** (common likers) แสดงรายชื่อ**คนกดไลค์** ตามลำดับตัวอักษร ที่กดไลค์ทั้ง **คนโพสต์1** และ **คนโพสต์2** ถ้าไม่มีให้แสดง **None**
4) `M` (find mutual likers) แสดง**คู่ของรายชื่อที่มีการกดไลค์ซึ่งกันและกันในรูปแบบทูเปิ้ล** ตามลำดับตัวอักษร เช่น A ไลค์ B และ B ก็ไลค์ A ก็จะแสดงสองทูเปิ้ลคือ `('A', 'B')` และ `('B', 'A')` ถ้าไม่มีให้แสดง **None**

หมายเหตุ ทุกคำสั่งให้เรียงชื่อตามลำดับพจนานุกรม (ดูตัวอย่าง)

ข้อมูลส่งออก

ข้อมูลส่งออกมีลักษณะจำเพาะกับแต่ละคำสั่ง (ดูตัวอย่าง)

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
`A Boy H D` `R T S D Ed` `Wii T Q S` `done` `R`	`A 3` `R 4` `Wii 3`
`D T A S D` `Wii T Q S` `H V D A F` `done` `T`	`A` `D` `S` `T`
`A Boy H D` `R T S D Ed` `X V D H F` `done` `C H D`	`A` `X`
`A H F` `X D F` `done` `C H D`	`None`
`A H F` `X D F` `done` `C X D`	`None`
`A Boy H D` `D T A S Ed` `H V D A F` `done` `M`	`('A', 'D')` `('A', 'H')` `('D', 'A')` `('H', 'A')`
`A Boy H D` `D T S Ed` `done` `M`	`None`

หมายเหตุ สีแดงคือ**ชื่อคนกดไลค์** สีม่วงคือ**ชื่อคนโพสต์** สีฟ้าคือ**คำสั่ง**

P-65: งูเหลือมจับงูเห่า

ประธานรัฐสภากะลาแลนด์จ้างคุณให้เขียนโปรแกรมเพื่อหาว่า ลูกพรรคคนใดโหวตไม่ตรงกับมติพรรค (เรียกว่า เป็นงูเห่า)

- มติพรรคคือ ตัวเลือกที่ลูกพรรคโหวตมากที่สุด
- การโหวตใด ๆ มีทั้งหมดสามตัวเลือก คือ รับ (**Y**) ไม่รับ (**N**) และไม่ลงคะแนน (**X**)
- กรณีที่มีผลโหวตอันดับสูงสุดในพรรคมากกว่าหนึ่งตัวเลือก หรือพรรคไม่มีคะแนนโหวตเลย ให้แสดงว่า **Inconclusive**

ข้อมูลนำเข้า

แบ่งเป็นสามส่วน

1. รายชื่อพรรค
2. รายชื่อ สส. และชื่อพรรคที่สังกัด
3. รายชื่อ สส. และผลโหวต ถ้า สส. ใดไม่อยู่ในลิสต์นี้ถือว่าขาดประชุม

สตริงชื่อพรรค และชื่อ สส.
ตัวอังกฤษใหญ่กับเล็กถือว่า ต่างกัน

ข้อมูลส่งออก

รายชื่อพรรค ตามด้วยรายชื่อ สส.ที่เป็นงูเห่าในแต่ละพรรค

- รายชื่อพรรคเรียงตามลำดับในรายชื่อพรรคที่อ่านเข้ามา
- รายชื่องูเห่าในแต่ละพรรคเรียงตามลำดับตัวอักษร
  - ถ้าไม่มีงูเห่า ให้แสดง **No cobra**
  - ถ้าสรุปมติพรรคไม่ได้ คือ ผลโหวตอันดับสูงสุดในพรรคมีมากกว่าหนึ่งตัวเลือก หรือพรรคไม่มีคะแนนโหวตเลย ให้แสดง **Inconclusive**

ตัวอย่าง

input (จากแป้นพิมพ์)	ผลโหวตแต่ละพรรค	output (ทางจอภาพ)
5 Illuminati Tangerine Grain Uncle Sq 20 IA Illuminati GB Grain GA Grain UA Uncle UZ Uncle UB Uncle UC Uncle UD Uncle UT Uncle GC Grain SA Sq SY Sq SB Sq TA Tangerine TB Tangerine UX Uncle US Uncle UI Uncle SD Sq SQ Sq	Illuminati: Y - 1, N - 0, X - 0 Tangerine: Y - 1, N - 1, X - 0 Grain: Y - 0, N - 1, X - 2 Uncle: Y - 2, N - 1, X - 1 Sq: Y - 1, N - 3, X - 1	Illuminati No cobra Tangerine Inconclusive Grain GC Uncle US, UX Sq SA, SY

รายชื่อพรรค

รายชื่อ สส. และพรรค

รายชื่อพรรคเรียงตามที่อ่านมา

รายชื่องูเห่าเรียงตามตัวอักษร

สตริงชื่อพรรค และชื่อ สส.
ตัวอังกฤษใหญ่กับเล็กถือว่า ต่างกัน

15 GB X GA X GC N IA Y SA X SY Y SB N TA Y TB N UI Y UB Y UX N US X SD N SQ N รายชื่อ สส . และโหวต		

P-66: เรตติ้งของการเล่นเกม

หนึ่งในสิ่งที่ใช้วัดความสามารถของผู้เล่นเกมแนว Music Game ได้คือ “Rating” ของผู้เล่น โดยทั่วไปแล้วถูกคำนวณมาจากคะแนนที่สามารถทำได้ในแต่ละเพลง ประกอบกับความยากของเพลง ในที่นี้ เราจะให้สูตรการคำนวณ Rating ของแต่ละเพลงของผู้เล่นเป็นไปตามสมการด้านล่าง (กรณีมีเศษให้ปัดลงให้เหลือเป็นจำนวนเต็ม)

$$Rating = 25 \times (SongLv + 1) \times \left(\frac{Score}{10^7}\right)$$

และ Rating รวมของผู้เล่นคนหนึ่ง จะได้จากการนำผลรวม Rating ของ 5 เพลงที่มี Rating สูงที่สุดมาบวกกัน (กรณีมีเพลงที่เคยเล่นไม่ถึง 5 เพลง ให้นำมาบวกกันเท่าที่มี)

จงเขียนโปรแกรมเพื่อบันทึกสถิติการเล่นเพลงต่าง ๆ และคำนวณ Rating ของแต่ละเพลง และของผู้เล่น

ข้อมูลนำเข้าและส่งออก

บรรทัดแรกเป็นจำนวนเต็ม N แทนจำนวนของคำสั่งที่จะได้รับ

จากนั้นอีก N บรรทัดเป็นหนึ่งในคำสั่งดังต่อไปนี้ (คำสั่งอาจตามด้วยข้อมูลประกอบ โดยคำสั่งและข้อมูลประกอบแต่ละตัวจะถูกคั่นด้วยเครื่องหมายขีดกลาง | มีรูปแบบตามที่ระบุด้านล่าง)

- `Play | Song name | Song Lv | Score` => บันทึกการเล่นเพลงดังกล่าว (กรณีมีประวัติการเล่นเพลงนี้แล้ว ให้บันทึกเฉพาะครั้งที่ได้ rating ดีกว่า หาก rating เท่ากัน ให้บันทึกครั้งที่มี Song Lv สูงกว่า และ Score สูงกว่าตามลำดับ)
- `Rating | Song name` => แสดงค่า rating ของเพลงดังกล่าว กรณีเป็นเพลงที่ไม่เคยเล่น ให้แสดงค่า `0`
- `Rating` => แสดงค่า rating รวมของผู้เล่น กรณีผู้เล่นไม่เคยเล่นเพลงอะไรเลย ให้แสดงค่า `0`
- `Detail | Song name` => แสดงข้อมูลเพลง `Song name | Song Lv | Score | Rating` กรณีเพลงดังกล่าวไม่เคยถูกเล่นมาก่อน ให้แสดง `Song name: No play history`

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
4 Play | Hello | 10 | 9000000 Play | Despacito | 15 | 9500000 Rating Rating | Hello	627 247
3 Play | Yuke | 18 | 9500000 Detail | Yuke Detail | Sirius	Yuke | 18 | 9500000 | 451 Sirius: No play history

P-67: หนึ่งมิตรชิดใกล้

<u>การคำนวณระยะห่างระหว่างเวกเตอร์</u>

ให้ $\mathbf{u} = \langle u_1, u_2, \dots, u_n \rangle$ และ $\mathbf{v} = \langle v, v_2, \dots, v_n \rangle$ เป็นเวกเตอร์ $(\mathbf{u}, \mathbf{v} \in \mathbb{R}^n)$

โจทย์นี้มีวิธีคำนวณ $distance(\mathbf{u}, \mathbf{v})$ ที่เป็นระยะห่างระหว่างเวกเตอร์ $\mathbf{u}$ และ$\mathbf{v}$ อยู่สามวิธี ดังนี้

1. Lp norm p เป็นจำนวนจริงใด ๆ ที่มากกว่า 0	$distance(\mathbf{u}, \mathbf{v}) = \left(\sum_{i=1}^{n} \lvert u_i - v_i \rvert^p\right)^{\left(\frac{1}{p}\right)}$, $p > 0$
2. Cosine distance (วัดทิศทางของเวกเตอร์)	$distance(\mathbf{u}, \mathbf{v}) = 1 - \frac{\sum_{i=1}^{n}(u_i v_i)}{\lVert\mathbf{u}\rVert \lVert\mathbf{v}\rVert}$ $\lVert\mathbf{u}\rVert = \sqrt{\sum_{i=1}^{n} {u_i}^2}$
3. Sign distance (นับจำนวนเครื่องหมายที่ต่างกัน)	$distance(\mathbf{u}, \mathbf{v}) = \sum_{i=1}^{n} diffsign(u_i, v_i)$ $sign(u_i) = \begin{cases} 1 & if\ u_i \geq 0 \\ 0 & if\ u_i < 0 \end{cases}$ $diffsign(u_i, v_i) = \begin{cases} 1 & if\ sign(u_i) \neq sign(v_i) \\ 0 & if\ sign(u_i) = sign(v_i) \end{cases}$

<u>สิ่งต้องการ</u>: ให้ $\mathbf{u}$ คือเวกเตอร์ และ $\mathbf{V}$ เป็นเมทริกซ์ที่แต่ละแถวมองเป็นเวกเตอร์ที่มีขนาดเดียวกับ $\mathbf{u}$

จงเขียนโปรแกรมที่หาเวกเตอร์ใน $\mathbf{V}$ ที่มีค่าระยะห่างน้อยที่สุดจากเวกเตอร์ $\mathbf{u}$ ตามวิธีคำนวณระยะห่างที่กำหนดให้

ข้อมูลนำเข้า

ข้อมูลนำเข้าจะรับจากแป้นพิมพ์ทีละบรรทัดดังนี้

- บรรทัดที่ 1: เวกเตอร์ $\mathbf{u}$ เป็นรายการของจำนวนคั่นด้วยเครื่องหมายจุลภาค
- บรรทัดที่ 2: ลักษณะ distance ที่จะใช้วัด มีสามแบบ (รับรองว่า มีสามแบบนี้เท่านั้น ไม่มีแบบอื่นแน่ ๆ)
  - **L*p*** (โดยที่ ***p*** เป็นตัวเลข เช่น **L1.5**)
  - **cos**
  - **sign**
- บรรทัดที่ 3: จำนวนเต็ม $\mathbf{k}$ คือ จำนวนแถวของเมทริกซ์ $\mathbf{V}$ ซึ่งแทนจำนวนเวกเตอร์ที่จะหาตัวที่ใกล้ที่สุดกับ $\mathbf{u}$
- อีก $\mathbf{k}$ บรรทัด บรรทัดละเวกเตอร์ซึ่งคือแต่ละแถวของ $\mathbf{V}$ แต่ละบรรทัดเป็นรายการของจำนวน คั่นด้วยจุลภาค

ข้อมูลส่งออก

เวกเตอร์ที่มีค่าระยะห่างน้อยที่สุด บอกเป็นตำแหน่ง (เริ่มที่ 0) ในกรณีที่เสมอ ให้คืนค่าเวกเตอร์เฉพาะเวกเตอร์ตำแหน่งแรกที่สุด

ตัวอย่าง

input (จากแป้นพิมพ์)	Distance	output (ทางจอภาพ)
1.0, 2.0, 3.0 L2.0 5 0, 5, 3 1.1, 2.1, 3.1 -1, -5.3, 4.5 -1, -5.3, 4.5 1.1, 2.1, 3.1	 3.16227766 0.17320508 7.71621669 7.71621669 0.17320508	1
1, -2 cos 4 1, -3 2, -4 -1, 2 0.5, 3.2	 0.01005051 0. 2. 1.81466539	1
1, -2, 3, -5, 0 sign 3 1, -2, 3, -5, -0.1 3, 1, 2, 5, 1 5, -10, 3, -2, 20	 1 2 0	2

ข้อกำหนดเพิ่มเติม

- ให้เริ่มโปรแกรมด้วยโค้ดด้านล่างนี้ ซึ่งเรียกใช้ฟังก์ชัน **readinput** เพื่ออ่านข้อมูลเข้ามาใช้ได้เลย

```
##################################################
## DON'T CHANGE ANY CODE in readinput
import numpy as np

def readinput():
    u = np.array([float(e) for e in input().split(',')])
    distance_type = input()
    num = int(input())
    V = np.zeros( (num, u.shape[0]) )
    for count in range(num):
        V[count,:] = np.array([float(e) for e in input().split(',')])
    return u, distance_type, V
##################################################

u, t, V = readinput()
```

- ห้ามใช้คำสั่งประเภทวงวน (for, while) ให้ใช้คำสั่ง numpy เพื่อหาคำตอบ

P-68: การแบ่งคำ

โจต้องการส่งข้อความหาแฟนสาวคนสวยชาวต่างประเทศของเขา เขาจึงมีความจำเป็นต้องพิมพ์ภาษาอังกฤษ แต่ปุ่ม space ดันกดไม่ได้ ทำให้โจต้องพิมพ์ตัวอักษรติดกัน โจต้องการโปรแกรมสำหรับการแบ่งคำเพื่อแก้ปัญหานี้ โจทย์ข้อนี้ให้เขียนโปรแกรมแบ่งคำเผื่อช่วยโจโดยแบ่งคำ โดยใช้วิธี longest matching ตามขั้นตอนดังนี้

1. ไล่ดูในสตริงจากอักขระตัวแรกจนถึงอักขระตัวสุดท้าย หาชุดอักขระติดกันที่ตรงกับคำที่อยู่ในพจนานุกรม
2. ให้เลือกตัดตามคำที่ยาวที่สุดที่พบในพจนานุกรมออกจากสตริง
3. ใช้สตริงที่เหลือ วนกลับไปทำข้อ 1 จนกว่าจะแบ่งไม่ได้แล้ว

** หากไม่พบในพจนานุกรม ให้ตัดรวมอักขระทั้งหมดที่แบ่งไม่ได้เป็นคำเดียวกัน **

ตัวอย่างการแบ่งคำ

สตริงที่ยังไม่ได้แบ่งคำ	คำในพจนานุกรม	สตริงผลลัพธ์
`"tappleisafood"`	`"app", "apple", "a", "at"`	`"t apple is a food"`

เริ่มดูตั้งแต่อักขระตัวแรกจาก **t** ไปจนถึงตัวสุดท้ายคือ **d** ซึ่งไม่พบคำในพจนานุกรมที่ขึ้นต้นด้วย **t** จึงแบ่ง **t** ออกมาจากข้อความตั้งต้น จากนั้นจึงเริ่มที่อักขระตัวถัดไปจาก **a** จนถึง **d** ตัวสุดท้าย จะได้ชุดของอักขระ **a, app, apple** ที่ตรงกับคำที่อยู่ในพจนานุกรมให้เลือกแบ่งเป็นคำที่ยาวที่สุดก็จะได้เป็น **apple** แบ่งออกมาจากข้อความตั้งต้น แล้วทำต่อโดยเริ่มที่อักขระตัวถัดไปคือ **i** ซึ่งไม่พบคำในพจนานุกรมที่ขึ้นต้นด้วย **i** จึงแบ่งออกไปเป็น **i** ถัดมาเริ่มที่ **s** ก็ไม่พบคำที่อยู่ในพจนานุกรมที่ขึ้นต้นด้วย **s** เช่นกัน ก็ให้รวม **i** กับ **s** เข้าด้วยกันกลายเป็น **is** ตัว **a** ถัดมามีอยู่ในพจนานุกรมและคำที่ยาวที่สุดแบ่งได้คือ **a** จึงแบ่ง **a** ออกมา สุดท้าย **food** อยู่ในพจนานุกรมเลยกลายเป็นคำเดียว

ข้อมูลนำเข้า

บรรทัดแรกเป็นข้อความที่ต้องการนำไปแบ่งคำ บรรทัดที่สองเป็นจำนวนคำในพจนานุกรม บรรทัดถัด ๆ ไปเป็นคำในพจนานุกรมบรรทัดละคำ ** หมายเหตุ ข้อความนำเข้าเป็นตัวพิมพ์เล็กทั้งหมด

ข้อมูลส่งออก

ข้อความผลลัพธ์หลังการแบ่งคำที่มีช่องว่างคั่นระหว่างคำ

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
thisisadog **8** **these** **this** **is** **a** **and** **his** **sad** **dog**	**this is a dog**
thisisscotchandhisdog **8** **these** **this** **is** **a** **and** **his** **sad** **dog**	**this is scotch and his dog**
thisiscocodog **2** **dog** **scotch**	**thisiscoco dog**

P-69: บอตค้าเหรียญคริปโต

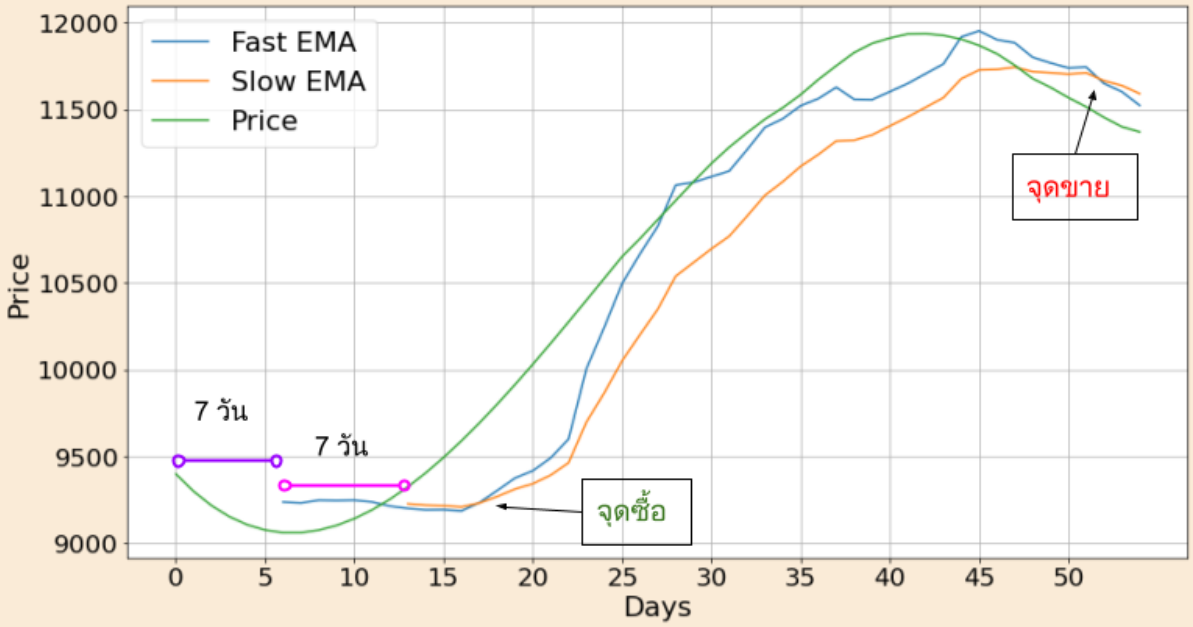
นายเม่ามองเห็นหนทางการทำกำไรในตลาดคริปโตแต่ขี้เกียจมานั่งเฝ้ากราฟดูราคา หลังจากศึกษามาสักระยะเวลานึง นายเม่าก็ได้ตัดสินสร้างบอทขึ้นมาช่วยในการเทรดโดยจะดูจุดซื้อจุดขายจากดรรชนี Exponential Moving Average (EMA) ที่มีสมการดังนี้

$$EMA(t_0) = \frac{1}{period}\sum_{i=0}^{period} price_i$$
$$EMA(t) = \alpha * price_t + EMA(t-1) * (1-\alpha)$$

กำหนดให้

$$\alpha = \frac{2}{1+period}$$

นายเม่าจะใช้ดรรชนี EMA ที่มีขนาด period ต่างกันสองตัว ได้แก่ Fast EMA และ Slow EMA ที่มีขนาด period เท่ากับ 7 และ 14 ตามลำดับ โดยหาจุดตัดระหว่าง Fast EMA และ Slow EMA หลังจากเส้นทั้งสองตัดกันแล้วหาก Fast EMA มีค่ามากกว่า Slow EMA จะนับเป็นจุดเข้าซื้อและถ้า Fast EMA มีค่าน้อยกว่า Slow EMA จะเป็นจุดขาย ดังตัวอย่างในรูปข้างล่างนี้



** ข้อสังเกต EMA จะเริ่มคำนวณวันแรก t_0 หลังจากผ่านไปตามจำนวน period ที่กำหนด เช่น EMA ที่มี period เท่ากับ 7 จะเริ่มคำนวณวันแรกในวันที่ 7 (นับจาก 1)

สิ่งที่โจทย์ต้องการ

1. รับข้อมูลนำเข้าซึ่งเป็นจำนวนสัปดาห์ และตามด้วยราคาของเหรียญ
2. ทำการหาจุดซื้อและขายด้วย Fast EMA และ Slow EMA
3. แสดงจุดซื้อจุดขายทั้งหมด หากไม่มีให้แสดงเป็น `No results`

** หมายเหตุ ทั้งหมดนี้เป็นเพียงโจทย์ ไม่แนะนำให้นำเอาไปใช้งานจริง

ข้อมูลนำเข้า

บรรทัดแรกเป็นจำนวนสัปดาห์ บรรทัดต่อ ๆ มาเป็นรายการราคาของแต่ละสัปดาห์ สัปดาห์ละหนึ่งบรรทัด ประกอบด้วยยรายการราคาของเหรียญ 7 วันคั่นด้วยเครื่องหมายจุลภาค

ข้อมูลส่งออก

สองบรรทัด แสดงรายการซื้อขายเป็นทศนิยมสองตำแหน่งตามรูปแบบดังนี้

{**BUY** หรือ **SELL**} **at** {ราคาของเหรียญ ณ วันที่ทำการซื้อขาย}

หากไม่มีให้แสดง **No results**

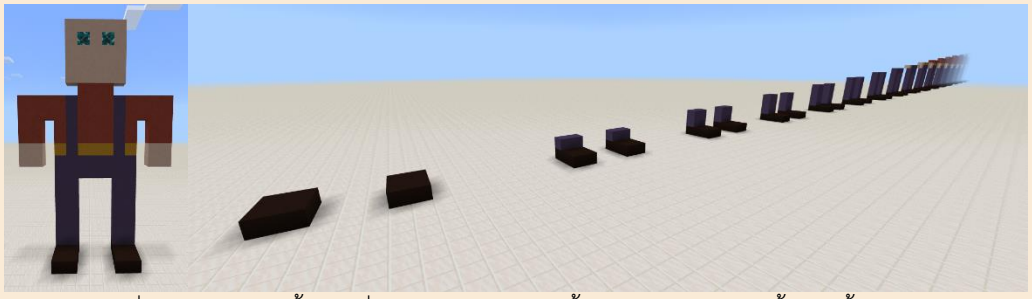
ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
8 9.13,9.08,9.29,9.25,9.45,9.24,9.23 9.22,9.29,9.24,9.25,9.21,9.14,9.16 9.17,9.2,9.17,9.38,9.51,9.59,9.54 9.71,9.93,11.22,10.96,11.25,11.18,11.31 11.77,11.13,11.21,11.24,11.64,11.78,11.59 11.75,11.69,11.82,11.35,11.55,11.74,11.79 11.88,11.93,12.38,12.06,11.75,11.83,11.55 11.67,11.65,11.76,11.37,11.46,11.29,11.47	**BUY at 9.38** **SELL at 11.37** ตัวอย่าง ค่า FastEMA กับ SlowEMA โดยประมาณใน 21 วันแรก FastEMA : X, X, X, X, X, X, 9.238 9.234, 9.248, 9.246, 9.247, 9.238, 9.213, 9.200 9.192, 9.194, 9.188, **9.236**, 9.305, 9.376, 9.417 SlowEMA : X, X, X, X, X, X, X X, X, X, X, X, X, 9.227 9.219, 9.217, 9.211, **9.233**, 9.270, 9.313
4 6.59,7.34,7.15,7.18,7.14,7.41,7.29 7.24,7.21,7.19,7.24,7.32,7.39,7.27 7.2,7.19,6.97,7.29,7.35,7.35,7.72 8.06,8.07,7.8,8.08,8.07,8.16,8.12	**SELL at 6.97** **BUY at 7.29**
1 34.99,31.04,32.85,32.05,32.15,32.35,32.84	**No results**

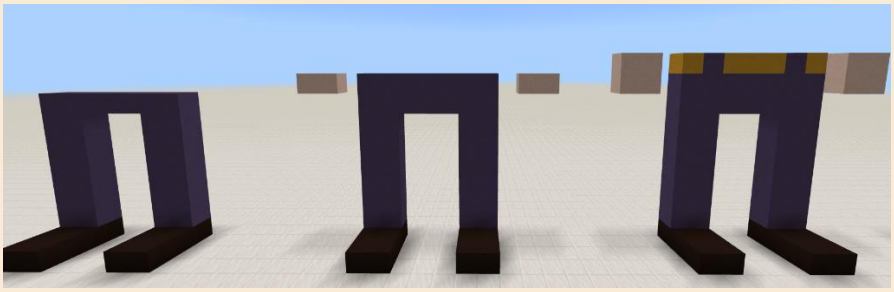
P-70: เกาะลอยอยู่ที่ใด

กระบวนการพิมพ์ชิ้นงานสามมิติ สามารถทำได้โดยการหั่นชิ้นงานเป็นชั้น ๆ บาง ๆ ตามแนวดิ่ง เครื่องพิมพ์จะค่อย ๆ พิมพ์ครั้งละ 1 ชั้นจากชั้นล่างสุด วางซ้อนทับเป็นชั้นใหม่ ซ้อนไปถึงชั้นบนสุด ได้จะชิ้นงานสามมิติออกมา

แต่ในกระบวนการซ้อนชั้นต่าง ๆ นั้น มีข้อจำกัดคือ หากชั้นปัจจุบันไม่มีชั้นล่าง จะไม่สามารถพิมพ์ได้ ตัวอย่างเช่น การพิมพ์ชิ้นงานดังแสดงในรูปที่ 1 ซ้าย จะพิมพ์ทีละชั้นดังแสดงในรูปที่ 1 ขวา และเกิดปัญหาในส่วนของมือ (รูปที่ 2) พลาสติกส่วนมือจะถูกฉีดลงไปติดกับพื้นระดับเดียวกับเท้า จึงต้องมีโครงสร้าง (เรียกว่า support) เพิ่มขึ้นมาเพื่อรองรับการพิมพ์ ไม่ให้พิมพ์ใส่อากาศ แต่พิมพ์ใส่ support นี้แทน เราเรียกส่วนมือที่ลอยอยู่นี้ว่า **Island**

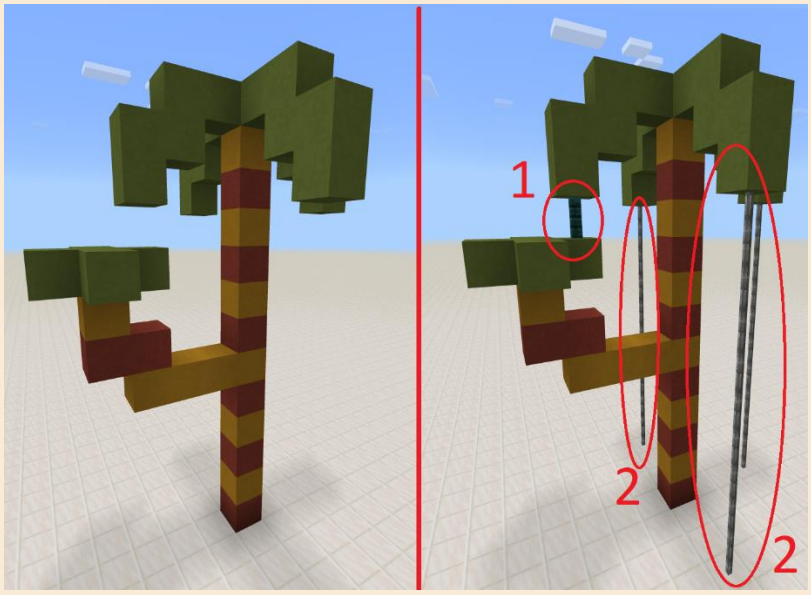


รูปที่ 1 (ซ้าย) แสดงชิ้นงานที่สมบูรณ์แล้ว, (ขวา) ขั้นตอนการพิมพ์ทีละชั้นจากชั้นล่างสุด



รูปที่ 2 ปัญหา island ที่เกิดขึ้นในส่วนมือ

Support มี 2 ประเภทคือ Main support และ Inner support (รูปที่ 3) Main support เป็น support ที่ฐานรับน้ำหนักมาจากที่พื้นเดียวกับชั้นแรกสุด แต่ Inner support เป็น support ที่มีฐานรับน้ำหนักเป็นเนื้อชิ้นงานชั้นต่ำกว่าที่อยู่ใกล้สุด

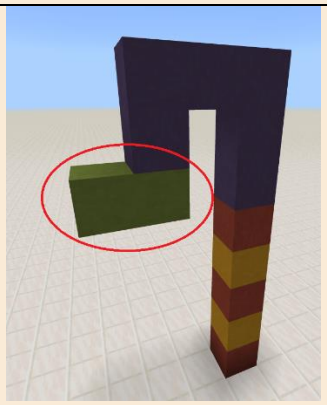
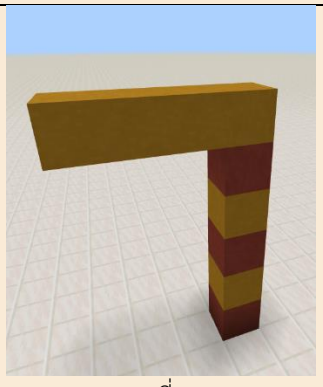
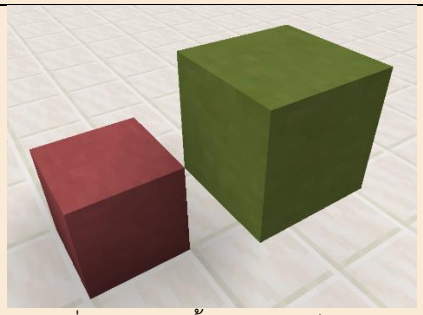


รูปที่ 3 ก่อนมี support (รูปซ้าย), หลังจากมี support แล้ว (รูปขวา) 1 คือ Inner support และ 2 คือ Main support

สิ่งที่ต้องทำในโจทย์นี้คือ

1. หาว่ามี Island เกิดขึ้นที่ใดบ้าง
2. หาว่า Island ที่เกิดขึ้นจะสามารถถูกปรับปรุงได้ด้วย Support แบบใด

คำถามที่พบบ่อย

Q: ในรูปที่ 4 หากมีหลายบล็อกที่ติดกัน ลอยอยู่ จะต้องตอบทุกบล็อกที่ติดกันหรือไม่ A: ใช่ครับ จะต้องตอบทุกบล็อกที่ติดกัน ตัวอย่างในรูป บล็อกสีเขียวทั้งสองอันจะลอยอยู่ ดังนั้นจะต้องตอบทั้ง 2 บล็อกครับ	 รูปที่ 4 ส่วนที่เป็น Island คือสีเขียวในวงกลมทั้งสองอัน
Q: ในรูปที่ 5 แบบนี้นับว่าเป็นเกาะหรือไม่ครับ A: ตัวอย่างนี้ไม่เป็นเกาะครับ เพราะว่าโครงสร้างด้านข้างของมันได้รับน้ำหนักให้แล้ว	 รูปที่ 5
Q: ทิศทางใดบ้างที่ถือว่าโครงสร้างได้รับน้ำหนักแล้ว A: เฉพาะ บน ล่าง ซ้าย และ ขวา เท่านั้น ในส่วนของทิศตามเส้นทะแยงมุมตัวอย่างในรูปที่ 6 ไม่จัดว่ารับน้ำหนักแล้วครับ	 รูปที่ 6 สีเขียว (ชิ้นบนขวา) เป็นเกาะ

ข้อมูลนำเข้า

- บรรทัดแรกเป็นตัวอักษร **`Y`** หรือ **`N`** บอกประเภทของคำถาม
  - ถ้ารับอินพุทเป็น **`N`** ให้ตอบเพียงตำแหน่งของ island,
  - ถ้ารับอินพุทเป็น **`Y`** ให้ตอบทั้ง ตำแหน่ง และ ประเภทของ support เป็นตัวอักษร **`I`** หรือ **`M`** แทน Inner support และ Main support ตามลำดับ
- บรรทัดที่สองบอกความกว้าง ความยาว และความสูง ถูกคั่นโดยเครื่องหมายจุลภาค **`W,L,H`**
- บรรทัดถัด ๆ มา (จำนวนบรรทัดเท่ากับ H คูณ L) เป็นตัวอักษร **`x`** หรือ **`-`** ติดกันจำนวน **`W`** ตัว ทุก ๆ **`L`** บรรทัดติดกันรวมกันเป็น 1 ชั้น โดยมีทั้งหมด **`H`** ชั้น โดย **`x`** แปลว่ามีเนื้อชิ้นงาน **`-`** แปลว่าไม่มีชิ้นงาน

ข้อมูลส่งออก

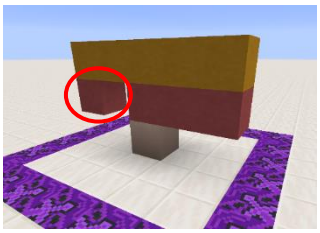
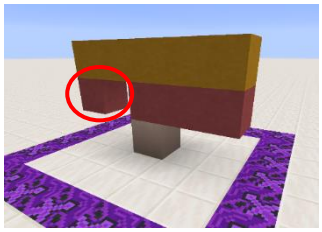
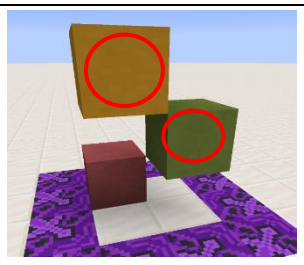
จำนวนบรรทัดเท่ากับจำนวน island ที่มี โดยการตอบต้องเรียงลำดับจาก island ใน layer ล่างสุดก่อน และตามด้วย ประเภทของ support (**I** ก่อน **M**) ตามด้วยแถว และสุดท้ายคือหลัก

ถ้าบรรทัดแรกของอินพุตคือ **N** แต่ละบรรทัดที่แสดงเป็นเลข 3 จำนวน คั่นด้วยจุลภาค ชั้นที่,แถวที่,คอลัมน์ที่ (L,X,Y) โดยชั้นจะเริ่มที่ 0

ถ้าบรรทัดแรกของอินพุตคือ **N** แต่ละบรรทัดที่แสดงเป็นเลข 4 จำนวน คั่นด้วยจุลภาค ชั้นที่,ประเภทของโครงสร้างรับน้ำหนัก,แถวที่, คอลัมน์ที่ (L,T,X,Y) โดยชั้นจะเริ่มที่ 0 และ ประเภทของโครงสร้างมีเพียง **I** และ **M** เท่านั้น

หากไม่มี island ให้ตอบว่า **`There is no island`** (ไม่มีเครื่องหมายอัญประกาศคู่)

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
N **3,3,1** **xx-** **x--** **x-** 	**`There is no island`**
N **5,5,3** **-----** **-----** **--x--** **-----** **-----** **--x--** **-----** **--x--** **--x--** **--x--** **--x--** **--x--** **--x--** **--x--** 	**1,0,2**
Y **5,5,3** **-----** **-----** **--x--** **-----** **-----** **--x--** **-----** **--x--** **--x--** **--x--** **--x--** **--x--** **--x--** **--x--** 	**1,M,0,2**
Y **2,2,3** **x-** **--** **--** **-x** **--** **x-** 	**1.M,1,1** **2,M,1,0**

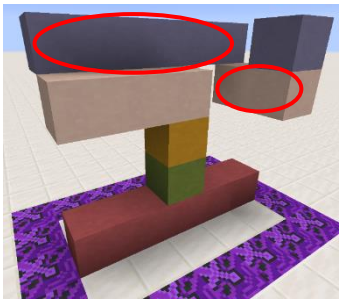
```
Y
5,3,5
-----
xxxxx
-----

-----
--x--
-----

-----
--x--
-----

-----
xxx-x
----x

xxxxx
-----
----x
```



```
3,I,1,4
3,M,2,4
4,M,0,0
4,M,0,1
4,M,0,2
4,M,0,3
4,M,0,4
```

โค้ดตั้งต้น

สำหรับข้อนี้จะมีการให้ฟังก์ชัน **to_cluster** ซึ่งรับข้อมูลเป็นข้อมูลของชั้นเพียงหนึ่งชั้นในรูปแบบของ numpy array 2 มิติ ซึ่งมีค่าเป็น **True** สำหรับเนื้อชิ้นงาน และ **False** หากไม่มีอะไร โดยฟังก์ชันนี้จะแยกเนื้อของชิ้นงานที่นับว่าเป็นชิ้นเดียวกันสำหรับชั้นนี้ให้ โดยคืนค่ากลับมาเป็นลิสต์ของ numpy array ที่มี dimension เหมือนข้อมูลนำเข้า

True	False	True
True	False	True
True	False	True

⇨

True	False	False
True	False	False
True	False	False

,

False	False	True
False	False	True
False	False	True

รูปด้านบนทางซ้ายมีอยู่สองกล่อง เมื่อเอาไปใส่ในฟังก์ชัน **to_cluster** ก็จะถูกแยกออกเป็นลิสต์สองช่อง แต่ละช่องเป็นอาเรย์

```
import numpy as np

def to_cluster(a):
  """
  Seperate cluster from 2d array.
  # Parameters
  a: array_like - 2 dimension array of `True` or `False` with dtype np.bool.

  # Returns
  cluster_list: List - a list of 2d array each with only one cluster.

  # Notes
  This function doesn't guarantee order of output.

  # Examples
  >>> a = np.array([
        [1, 0, 1],
        [1, 0, 1],
        [1, 0, 1],
      ], np.bool)
  >>> np.array(to_cluster(a))

  array([[[ True, False, False],
      [ True, False, False],
      [ True, False, False]],
       [[False, False,  True],
      [False, False,  True],
      [False, False,  True]]])
  """
```

```
    height, width = a.shape
    output = []
    last_cross_section = []
    start_row = np.argmax(np.pad(np.max(a, axis=1), ((0, 1),),
                                 mode='constant', constant_values=np.inf))
    for row_index in range(start_row, height):
      cross_section = []
      seg_start = np.argmax(np.pad(a[row_index], ((0, 1),), mode='constant',
                                   constant_values=np.inf))
      for col_index in range(seg_start, width + 1):
        if (col_index == width and a[row_index][col_index-1]) or \
           (col_index < width and a[row_index][col_index-1] and \
            not a[row_index][col_index]):
          seg = np.full_like(a, False)
          seg[row_index, seg_start:col_index] = True
          cross_section.append(seg)
        if col_index < width and not a[row_index][col_index-1] and \
           a[row_index][col_index]:
          seg_start = col_index
      marked_remove = set()
      for lcs in last_cross_section:
        merged_to = None
        for ic, ccs in enumerate(cross_section):
          if np.any(np.logical_and(lcs[row_index - 1], ccs[row_index])):
            if merged_to is None:
              ccs[:, :] = np.logical_or(lcs, ccs)  # inplace merge
              merged_to = ccs
            else:  # merge into the same lcs, so need to remove from last_cross_section
              merged_to[:, :] = np.logical_or(merged_to, ccs)  # inplace merge
              marked_remove.add(ic)
        if merged_to is None:
          output.append(lcs)
      last_cross_section = [cross_section[ic] for ic in range(len(cross_section))
                            if ic not in marked_remove]
    output.extend(last_cross_section)
    return output
```

R-71: ฟังก์ชันเวียนบังเกิด

จงเขียนฟังก์ชันตามนิยามที่กำหนดให้ ดังต่อไปนี้

Tower of Hanoi	`def h(n):`	$h(n) = 2h(n-1) + 1 \ \ if \ n \geq 1, \ \ h(0) = 0$
Greater Common Divisor	`def gcd(x,y):`	$gcd(x,y) = gcd(y, x \bmod y) \ \ if \ y > 0, \ \ \ gcd(x,0) = x$
Josephus Problem	`def J(n,k):`	$J(n,k) = (J(n-1,k) + k) \bmod n \ \ if \ n > 1, \ \ J(1,k) = 0$
Catalan Number	`def C(n):`	$C(n+1) = \sum_{k=0}^{n} C(k)C(n-k) \ \ if \ n \geq 0, \ \ C(0) = 1$
Fibonacci Number	`def f(n):`	$f_{2n-1} = f_n^2 + f_{n-1}^2 \ \ if \ n \geq 2$ $f_{2n} = (2f_{n-1} + f_n)f_n \ \ if \ n \geq 1$ $f_0 = 0, \ \ f_1 = 1$
Hofstadter Female and Male sequences	`def F(n):` `def M(n):`	$F(n) = n - M\big(F(n-1)\big) \ if \ n > 0$ $M(n) = n - F\big(M(n-1)\big) \ if \ n > 0$ $F(0) = 1, \ \ M(0) = 0$
Ackermann Number	`def A(m,n):`	$A(m,n) = \begin{cases} A(m-1,1) & if \ m > 0 \ and \ n = 0 \\ A(m-1, A(m,n-1)) & if \ m > 0 \ and \ n > 0 \\ n+1 & if \ m = 0 \end{cases}$

เขียนฟังก์ชัน ในโครงของโปรแกรมข้างล่างนี้

```
def h(n):       # Tower of Hanoi

def gcd(x,y):   # Greatest Common Divisor

def J(n,k):     # Josephus Problem

def C(n):       # Catalan Number

def f(n):       # Fibonacci Number

def F(n):       # Female sequence

def M(n):       # Male sequence

def A(m,n):     # Ackermann Number


exec(input().strip()) # do not remove this line
```

ข้อมูลนำเข้า

คำสั่งในการทดสอบฟังก์ชันที่เขียน

ข้อมูลส่งออก

ผลที่ได้จากคำสั่งที่ป้อนเป็นข้อมูลนำเข้า

ตัวอย่าง

input (ทางแป้นพิมพ์)	output (ทางจอภาพ)
`print(h(15))`	`32767`
`print(gcd(60,81))`	`3`
`print(J(30,5))`	`2`
`print(C(15))`	`9694845`
`print(f(80))`	`23416728348467685`
`print(F(40))`	`25`
`print(M(50))`	`31`
`print(A(3,2))`	`29`

R-72: ฟังก์ชันเวียนบังเกิด (ยังมีอีก)

จากฟังก์ชันตามนิยามที่กำหนดให้ต่อไปนี้ จงเขียนฟังก์ชันในโครงของโปรแกรมข้างล่างนี้

Logistic Map	`def x(n):`	$x_{n+1} = 3x_n(1-x_n)\ \ if\ n \geq 0,\ \ x_0 = 0.01$
Motzkin number	`def M(n):`	$M_n = M_{n-1} + \sum_{k=0}^{n-2} M_k M_{n-2-k}\ \ if\ n \geq 2,\ \ M_0 = 1, M_1 = 1$
Delannoy number	`def D(m,n):`	$D(m,n) = D(m-1,n) + D(m-1,n-1) + D(m,n-1)\ \ if\ \ m,n > 0,$ $D(m,0) = D(0,n) = 1$
Schröder–Hipparchus number	`def S(n):`	$S(n) = \frac{1}{n}\big((6n-9)S(n-1) - (n-3)S(n-2)\big)\ \ if\ n \geq 3,$ $S(1) = S(2) = 1$
Derangement	`def d(n):`	$d_n = nd_{n-1} + (-1)^n\ if\ n \geq 1,\ \ x_0 = 1$

```
def x(n):       # Logistic Map

def M(n):       # Motzkin number

def D(m,n):     # Delannoy number

def S(n):       # Schroder-Hipparchus number

def d(n):       # Number of Derangements


exec(input().strip()) # do not remove this line
```

ข้อมูลนำเข้า

คำสั่งในการทดสอบฟังก์ชันที่เขียน

ข้อมูลส่งออก

ผลที่ได้จากคำสั่งที่ป้อนเป็นข้อมูลนำเข้า

ตัวอย่าง

input (ทางแป้นพิมพ์)	output (ทางจอภาพ)
`print(x(5))`	`0.23805842983255365`
`print(M(5))`	`21`
`print(D(5,5))`	`1683`
`print(S(5))`	`45`
`print(d(5))`	`44`

R-73: การเรียงลำดับแบบ quicksort

การเรียงลำดับข้อมูล **(sorting)** มีหลายวิธี โจทย์นี้นำเสนอการ **sort** แบบหนึ่งเรียกว่า **qsort** มีหลักการทำงานแบบ **recursive** ดังนี้

```
import random

def qsort( d ):
    _____________ # ถ้า d มีข้อมูลไม่เกิน 1 ตัว ก็คืน d กลับไปได้เลย เพราะเรียงอยู่แล้ว
    p = d[random.randint(0,len(d)-1)]  # สุ่มเลือกข้อมูลใน d มาหนึ่งตัว เก็บในตัวแปร p
    _____________ # สร้าง list ชื่อ le เก็บข้อมูลใน d ทุกตัวที่มีค่าน้อยกว่า p (เขียนด้วย list comprehension)
    _____________ # สร้าง list ชื่อ eq เก็บข้อมูลใน d ทุกตัวที่มีค่าเท่ากับ p (เขียนด้วย list comprehension)
    _____________ # สร้าง list ชื่อ mo เก็บข้อมูลใน d ทุกตัวที่มีค่ามากกว่า p (เขียนด้วย list comprehension)
    _____________ # เรียงลำดับข้อมูลใน le ด้วย qsort เก็บผลใส่ le
    _____________ # เรียงลำดับข้อมูลใน mo ด้วย qsort เก็บผลใส่ mo
    _____________ # เมื่อนำ le ต่อกับ eq ต่อกับ mo จะได้ข้อมูลเรียงจากน้อยไปมาก คืนผลการต่อนี้กลับเป็นผลลัพธ์

d = [int(e) for e in input().split()]
d = qsort(d)
print(' '.join([str(e) for e in d]))
```

จงเติมคำสั่งในฟังก์ชัน **qsort** ข้างบนนี้ให้ถูกต้องตาม **comment** ที่เขียนกำกับแต่ละบรรทัด

ข้อมูลนำเข้า

รายการของจำนวนเต็มคั่นด้วยช่องว่าง 1 บรรทัด

ข้อมูลส่งออก

ผลลัพธ์จากการเรียงลำดับข้อมูลที่ได้รับในข้อมูลนำเข้า

ตัวอย่าง

input (ทางแป้นพิมพ์)	Output (ทางจอภาพ)
87	87
1 2 3	1 2 3
3 2 1	1 2 3
1 2 1 2 1 2 1 2 1 2	1 1 1 1 1 2 2 2 2 2

R-74: เปลี่ยนฐานสิบเป็นฐานสิบหก

ฟังก์ชัน **dec2bin(d)** ข้างล่างนี้รับจำนวนเต็ม **d** เพื่อแปลงเป็น string ที่มีค่า 0 กับ 1 ซึ่งแทนจำนวนฐานสองของค่า **d** เช่น ถ้า **d = 30** การเรียก **dec2bin(d)** จะได้ string **"11101"** เป็นผลลัพธ์

```
def dec2bin(d):
    if d < 2 : return str(d)
    return dec2bin(d//2) + dec2bin(d%2)
```

จงใช้แนวคิดการทำงานของฟังก์ชัน **dec2bin(d)** มาเขียนฟังก์ชัน **dec2hex(d)** ซึ่งทำงานแบบ recursive เพื่อแปลงจำนวนเต็ม **d** เป็น string ที่ประกอบด้วย 0, 1, 2, 3, ..., 9, A, B, C, D, E, F ซึ่งแทนจำนวนฐานสิบหกของค่า **d** โดยใช้โครงของโปรแกรมข้างล่างนี้

```
def dec2hex(d):

  ???

n = int(input())
print(dec2hex(n))
```

ข้อมูลนำเข้า

จำนวนเต็ม **n** (**0 ≤ n ≤ 1000000000000000**)

ข้อมูลส่งออก

สตริงที่แทนการเขียนจำนวน **n** ในฐานสิบหก

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
0	0
1	1
9	9
10	A
15	F
31	1F
100	64
9999	270F
1029301932312	EFA72D1118

R-75: การหาจำนวนฟิโบนักชีอย่างรวดเร็ว

0, 1, 1, 2, 3, 5, 8, ... เป็นลำดับของจำนวนฟิโบนักชี ($F_0 = 0, F_1 = 1, F_2 = 1, ...$) วิธีหนึ่งในการหา F_n คือคำนวณ $\begin{bmatrix}0 & 1\\ 1 & 1\end{bmatrix}^n$ ได้ผลเป็นเมทริกซ์ขนาด 2×2 มี F_n ที่มุมขวาบนของเมทริกซ์ เช่น $\begin{bmatrix}0 & 1\\ 1 & 1\end{bmatrix}^3 = \begin{bmatrix}1 & 2\\ 2 & 3\end{bmatrix}$, $\begin{bmatrix}0 & 1\\ 1 & 1\end{bmatrix}^4 = \begin{bmatrix}2 & 3\\ 3 & 5\end{bmatrix}$

ถ้าคิดดูดี ๆ จะพบว่าการหาด้วยวิธีข้างต้นนี้คือการหาค่ายกกำลัง ซึ่งเราก็ไม่น่าหาแบบค่อยๆ คูณไปทีละครั้ง คือถ้า A เป็นเมทริกซ์ การหา A^{10} ก็ไม่น่าใช้วิธีที่เริ่มด้วยเมทริกซ์เอกลักษณ์ I = $\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}$ แล้วคูณด้วย A ไป 10 ครั้ง น่าจะใช้วิธีการหา A^5 แล้วจับมาคูณกับตัวเอง ก็จะได้ A^{10} เหมือนกับที่เรียนเรื่อง power mod นั่นคือ

$$A^n = \begin{cases} \mathrm{I} & n = 0 \\ \left(A^{\lfloor n/2 \rfloor}\right)^2 & n \text{ is even} \\ A\left(A^{\lfloor n/2 \rfloor}\right)^2 & n \text{ is odd} \end{cases}$$

จงเขียนฟังก์ชัน **fib(n, k)** เพื่อคำนวณ F_n **% k** ด้วยวิธีข้างต้นนี้ โดยใช้ คำสั่งใช้ numpy เพื่อการคูณเมทริกซ์

(หมายเหตุ : หลังการคูณเมทริกซ์ทุกครั้ง ให้นำผลที่ได้มา % k numpy จะทำ % k แบบ element-wise ในเมทริกซ์)

```
import numpy as np

def fib(n, k):

   ???

n,k = [int(e) for e in input().split()]
print( fib(n,k) )
```

ข้อมูลนำเข้า

จำนวนเต็ม 2 ค่า **n** กับ **k** (**0 ≤ n ≤ 10000000000000, 0 ≤ k ≤ 100000**)

ข้อมูลส่งออก

แสดงค่า F_n **% k**

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
0 10	0
1 10	1
2 10	1
89 10	9
11111 111	55
1111111111 111	76
1234567890 1234	162
10000000000000 999	600

R-76: การค้นข้อมูลในทูเปิลหลาย ๆ ชั้น

เราเก็บข้อมูลใน tuple แบบ tuple ซ้อน ๆ อยู่ใน tuple ได้ เช่น **`x = (4,(3,(5,6)))`**
แต่ถ้าเราจะหาข้อมูลใน tuple แบบนี้ ด้วยคำสั่ง in อาจหาไม่พบ เช่น **`4 in x`** จะได้ **`True`** แต่ **`3 in x`** จะได้ **`False`** เพราะการค้นแบบ **`in x`** จะหยิบข้อมูลใน **`x`** ทีละตัวมาเปรียบเทียบ คือ **`4`** และ **`(3,(5,6))`** ซึ่งไม่เท่ากับ **`3`**

โจทย์ข้อนี้ให้เขียนฟังก์ชัน **`find_in_nested_tuple (e,x)`** ที่คืน **`True`** ถ้ามี **`e`** ในระดับใดก็ได้ใน tuple **`x`** และคืน **`False`** ถ้าหาไม่พบ วิธีหนึ่งในเขียนฟังก์ชันนี้คือ เขียนแบบ recursive ตามโครงของโปรแกรมข้างล่างนี้ (ไม่ต้องแก้ไข 6 บรรทัดสุดท้าย เพราะเป็นคำสั่งอ่านข้อมูลเข้ามาทดสอบฟังก์ชันและแสดงผลลัพธ์)

```
def find_in_nested_tuple(e, x):




x = eval(input())
es = [int(e) for e in input().split()]
results = []
for e in es:
    results.append(find_in_nested_tuple(e,x))
print(" ".join([str(e) for e in results]))
```

หมายเหตุ : คำสั่ง **`type(b) is tuple`** จะคืน **`True`** ถ้าตัวแปร **`b`** เก็บ tuple ไม่เช่นนั้นคืน **`False`**

ข้อมูลนำเข้า

บรรทัดแรกคือ tuple ที่จะใช้ทดสอบ
บรรทัดที่สองเป็นรายการของจำนวนเต็มที่ต้องการนำไปค้นใน tuple ที่รับให้บรรทัดที่แล้ว

ข้อมูลส่งออก

รายการผลลัพธ์ของการค้นข้อมูลตามที่กำหนดในข้อมูลขาเข้า

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
`()` `0 1`	`False False`
`(1,2,3)` `0 1 2 3 4`	`False True True True False`
`(1,(2,(3,)),4)` `0 1 2 3 4 5`	`False True True True True False`
`(1,(2,()),4)` `0 1 2 3 4 5`	`False True True False True False`

R-77: คลี่ดิกหลายชั้นให้เหลือชั้นเดียว

โจทย์นี้ให้เขียนฟังก์ชัน **flatten_dict(d)** โดย **d** คือ dict ที่อาจมี value ของ key ใน dict เป็นอีก dict ซ้อน ๆ กันหลาย ๆ ชั้น เช่น **{ 'a':1, 'b':{'c':9, 'd':{'x':11, 'y':12}} }** หากเราจะ flatten dict ก็นำ value ในชั้นในๆ ออกมาอยู่นอกสุด โดยให้รวม key ที่ซ้อนๆ กันมาประกอบกันเป็น key ใหม่ เช่น จาก dict ข้างต้น เมื่อ flatten แล้วได้ **{ 'a':1, 'b.c':9, 'b.d.x':11, 'b.d.y':12 }**

จงเขียนฟังก์ชัน **flatten_dict(d)** ตามโครงของโปรแกรมข้างล่างนี้
(2 บรรทัดสุดท้ายอ่าน input เพื่อทดสอบฟังก์ชัน และแสดงผลลัพธ์ ไม่ต้องแก้ไขบรรทัดเหล่านี้)

```
def flatten_dict(d):




for k in sorted(flatten_dict(eval(input()))):
    print(k,':', x[k])
```

หมายเหตุ : คำสั่ง **type(x) is dict** คืน **True** เมื่อ **x** เป็น dict และคืน **False** ถ้า **x** ไม่ใช่ dict

ข้อมูลนำเข้า

สตริงหนึ่งบรรทัดแทน dict ที่จะใช้ทดสอบ dict นี้ และที่ซ้อน ๆ อยู่ จะมี key เป็น string หมด

ข้อมูลส่งออก

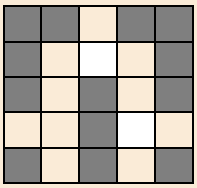
ข้อมูล key ตามด้วย value ของ dict ที่ถูก flatten แล้ว บรรทัดละ key-value โดยเรียงตาม key จากค่าน้อยไปมาก ให้ใช้คำสั่งที่เขียนให้ในโครงของโปรแกรมข้างบนนี้

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
`{'a':1, 'b':2, 'c':3}`	`a : 1` `b : 2` `c : 3`
`{'a':1, 'b':{'c':9, 'd':{'x':11, 'y':12}}}`	`a : 1` `b.c : 9` `b.d.x : 11` `b.d.y : 12`

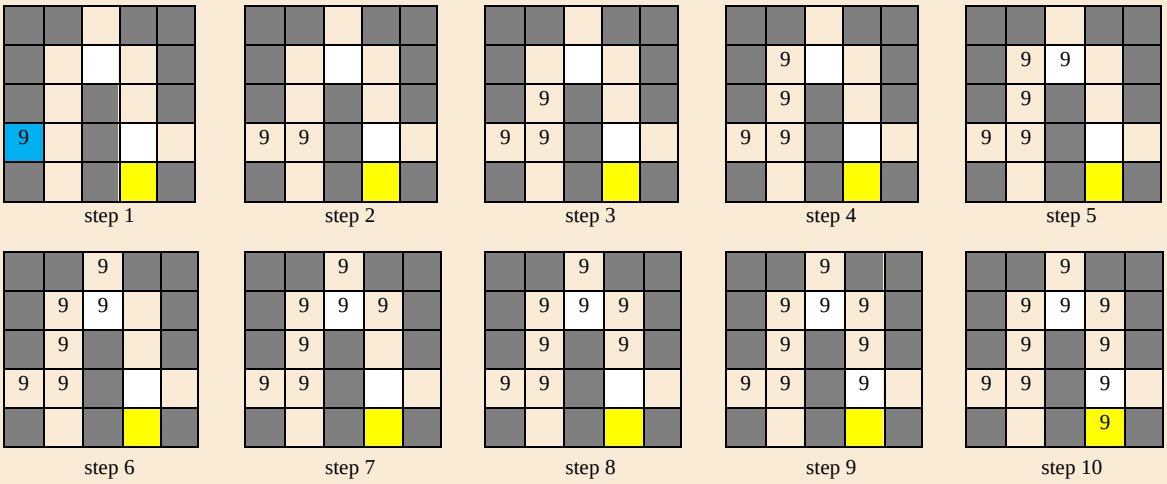
R-78: การหาวิถี

รูปทางซ้ายข้างล่างนี้แทนพื้นที่ในห้อง ๆ หนึ่งที่แบ่งออกเป็นช่อง ๆ เหมือนตาราง สีขาวแทนที่ว่าง สีเทาแทนสิ่งกีดขวาง เราสามารถแทนพื้นที่ในห้องนี้ด้วย list of lists ดังรูปทางขวาข้างล่างนี้ ให้ 0 แทนช่องว่าง 1 แทนสิ่งกีดขวาง



```
m = [ [ 1, 1, 0, 1, 1 ],
      [ 1, 0, 0, 0, 1 ],
      [ 1, 0, 1, 0, 1 ],
      [ 0, 0, 1, 0, 0 ],
      [ 1, 0, 1, 0, 1 ] ]
```

หากเราต้องการหาทางเดินจากช่อง (3,0) ช่องฟ้าข้างล่างนี้ ไปยังช่อง (4,3) ช่องเหลือง ก็ค่อย ๆ เติมเลขอะไรก็ได้ที่ไม่ใช่ 0 ในตาราง เริ่มจากช่องเริ่มต้นไปเรื่อย ๆ จนพบ (หรือไม่พบ) ช่องปลายทาง ดังแสดงในรูปข้างล่างนี้



สิ่งที่จะให้เขียนคือ ฟังก์ชัน **findPath(m,r,c,tr,tc)** มีหน้าที่หาว่าในตาราง **m** มีทางเดินจากตำแหน่ง แถว **r**, คอลัมน์ **c** ไปยังแถว **tr**, คอลัมน์ **tc** หรือไม่ (ถ้ามีคืน **True** ไม่มีคืน **False**) เราเขียนฟังก์ชันนี้แบบ recursive ตามที่โครงโปรแกรมที่เขียนข้างล่างนี้ (6 บรรทัดสุดท้ายเป็นการรับข้อมูลและแสดงผล ไม่ต้องแก้ไขใด ๆ)

```
import copy
def findPath(m, r, c, tr, tc):









nrows = int(input())
m = [[int(e) for e in input().split()] for i in range(nrows)]
nquestions = int(input())
for k in range(nquestions):
    sr,sc,tr,tc = [int(e) for e in input().split()]
    print(findPath(copy.deepcopy(m),sr,sc,tr,tc))
```

ข้อมูลนำเข้า

บรรทัดแรก จำนวนเต็ม **nrows** แทนจำนวนแถวของตาราง

nrows บรรทัดต่อมา แต่ละบรรทัดมีจำนวนเต็มที่มีจำนวนเท่ากัน แทนตาราง **m**

บรรทัดต่อมาเป็นจำนวนเต็ม **nquestions** แทนจำนวนคำถามที่จะตามมา

nquestions บรรทัดต่อมา แต่ละบรรทัดเป็นจำนวนเต็ม 4 จำนวน **sr, sc, tr, tc** เพื่อถามว่ามีทางเดินใน **m** เริ่มที่ตำแหน่งแถว **sr**, คอลัมน์ **sc** ไปจนถึงตำแหน่งแถว **tr**, คอลัมน์ **tc** หรือไม่

ข้อมูลส่งออก

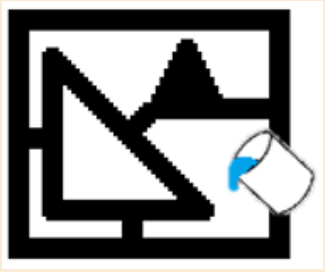
ผลลัพธ์ของการค้นทางเดิน (เป็น **`True`** หรือ **`False`**) จำนวน **`k`** ตัว ตามจำนวนคำถามในข้อมูลขาเข้า

ตัวอย่าง

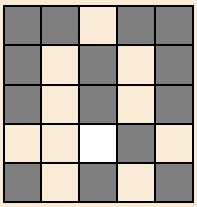
input (จากแป้นพิมพ์)	output (ทางจอภาพ)
5 1 1 0 1 1 1 0 0 0 1 1 0 1 1 1 0 0 1 0 0 1 0 1 0 1 4 0 2 4 1 0 2 4 3 0 2 3 0 4 3 3 4	True False True True

R-79: การเทสีในบริเวณปิด

นิสิตคงเคยใช้โปรแกรมวาดรูปที่เราสามารถเติมสีลงไปในช่องว่าง ดังแสดงตัวอย่างในรูปข้างล่างนี้



ภาพในคอมพิวเตอร์เกิดจากการเรียงจุดมองได้เป็นตาราง 2 มิติ ที่แต่ละช่องเก็บเลขสี เราจึงแทนภาพได้ด้วย list of lists เช่น ภาพทางซ้ายข้างล่างนี้มีขนาด 5 × 5 จุด แทนได้ด้วย list **x** ทางขวา โดย **1** แทนสีเทา **0** แทนสีขาว



```
x = [ [ 1, 1, 0, 1, 1 ],
      [ 1, 0, 1, 0, 1 ],
      [ 1, 0, 1, 0, 1 ],
      [ 0, 0, 0, 1, 0 ],
      [ 1, 0, 1, 0, 1 ] ]
```

สิ่งที่จะให้เขียนคือ ฟังก์ชัน **floodfill(x, row, col, c)** มีหน้าที่เติมค่า **c** เริ่มที่ช่องแถวแนวนอนที่ **row** และแถวแนวตั้งที่ **col** และแพร่ไปช่องอื่น ๆ ข้างเคียงไปเรื่อยจนเติมไม่ได้ เช่น หากเราเรียก **floodfill(x, 3, 1, 9)** จะทำให้ **x** กลายเป็น

```
x = [ [ 1, 1, 0, 1, 1 ],
      [ 1, 9, 1, 0, 1 ],
      [ 1, 9, 1, 0, 1 ],
      [ 9, 9, 9, 1, 0 ],
      [ 1, 9, 1, 0, 1 ] ]
```

โดยใช้โครงของโปรแกรมข้างล่างนี้ (6 บรรทัดสุดท้ายเป็นการรับข้อมูลและแสดงผล ไม่ต้องแก้ไขใด ๆ)

(เพื่อความง่าย **floodfill** จะเทสีให้กับสี **0** เท่านั้น และสีจะกระจายไปจนถึงขอบภาพหรือถึงบริเวณที่ไม่ใช่ **0**)

```
def floodfill(x, row, col, c):




nrows = int(input())
x = [[int(e) for e in input().split()] for i in range(nrows)]
row,col = [int(e) for e in input().split()]
floodfill(x, row, col, 9)
for each_row in x :
  print( " ".join([str(e) for e in each_row]) )
```

ข้อมูลนำเข้า

บรรทัดแรก จำนวนเต็ม **nrows** แทนจำนวนแถว

nrows บรรทัดต่อมา แต่ละบรรทัดมีจำนวนเต็มที่มีจำนวนเท่ากัน

บรรทัดสุดท้ายเป็นจำนวนเต็มสองจำนวนแทนตำแหน่งแถวและคอลัมน์เริ่มต้นที่จะทำ flood fill

ข้อมูลส่งออก

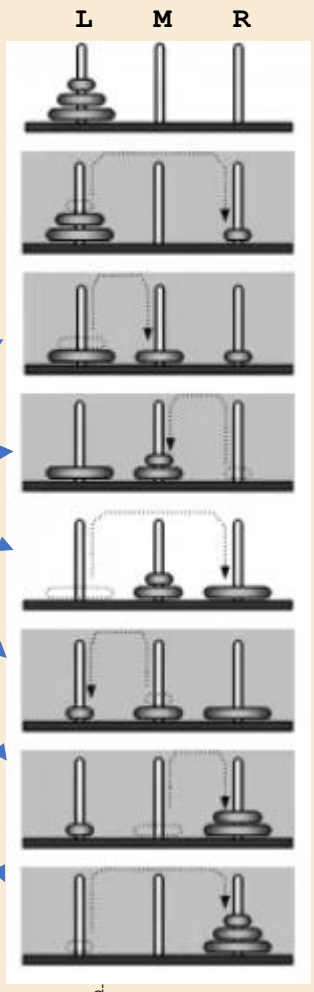
ข้อมูลใน list หลังการทำ flood fill แสดงบรรทัดละแถว

ตัวอย่าง

input (จากแป้นพิมพ์)	output (ทางจอภาพ)
3 1 1 1 1 0 1 0 0 0 2 1	1 1 1 1 9 1 9 9 9
5 1 1 0 1 1 1 0 1 0 1 1 0 1 0 1 0 0 0 1 0 1 0 1 0 1 3 1	1 1 0 1 1 1 9 1 0 1 1 9 1 0 1 9 9 9 1 0 1 9 1 0 1

R-80: หอคอยฮานอย

มีเสาอยู่ 3 ต้น จานกลมอยู่ 3 ใบมีรูตรงกลางขนาดไม่เท่ากัน เสียบในเสาต้นซ้าย เรียงจานใบเล็กวางทับอยู่บนใบใหญ่กว่าจากบนลงล่าง (รูปที่ 1 บนสุด) ปัญหาคือ จะย้ายจานทั้ง 3 ใบอย่างไร จากเสาซ้ายมายังเสาขวา (รูปที่ 1 ล่างสุด) เงื่อนไขคือ ย้ายได้ทีละใบจากเสาต้นหนึ่งไปอีกต้นหนึ่ง และใบใหญ่ห้ามทับใบเล็ก (การย้ายที่ใช้จำนวนครั้งน้อยสุดคือ ย้ายจานดังลำดับภาพในรูปที่ 1)



รูปที่ 1 Tower of Hanoi

ให้จานแต่ละใบมีหมายเลขตามขนาดจาน (ใบเล็กเลข 1) โปรแกรมข้างล่างนี้ ใช้ฟังก์ชัน **hanoi** เพื่อแสดงลำดับการย้ายจานตามต้องการ (โดยใช้ **hanoi** ใช้ได้กับกรณีจาน **n** ใบใดๆ **n ≥ 0**)

```
def hanoi(n,left,mid,right):
  if n == 0 : return
  hanoi(n-1,left,right,mid)
  print(n,':',left,'->',right)
  hanoi(n-1,mid,left,right)

hanoi(3,'L','M','R') # ได้ผลแสดงทางขวานี้
```

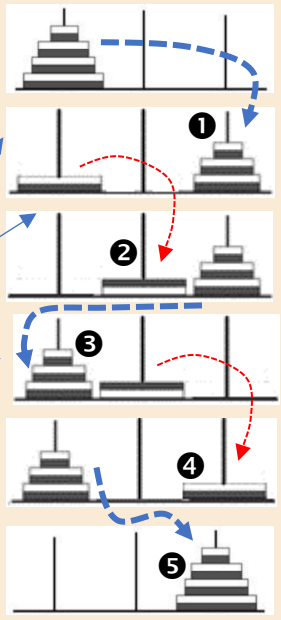
```
1 : L -> R
2 : L -> M
1 : R -> M
3 : L -> R
1 : M -> L
2 : M -> R
1 : L -> R
```

หน้าที่ของ **hanoi(n,left,mid,right)** นี้คือ แสดงวิธีย้ายจานหมายเลข 1 ถึง **n** จากเสา **left** ไปเสา **right** โดยใช้เสา **mid** เป็นเสาชั่วคราวระหว่างการย้าย มีการทำงานคือ

1. ถ้า **n** เป็น 0 แสดงว่าไม่มีจานให้ย้าย ก็ไม่ต้องทำอะไร แต่ถ้า **n > 0** ก็ทำขั้นตอนต่อไป
2. ทำ **hanoi(n-1,left,right,mid)** คือย้ายจานหมายเลข 1 ถึง **n-1** จาก **left** ไป **mid** ให้ได้ โดยใบใหญ่สุดหมายเลข **n** อยู่ที่เสา **left** ตลอดเวลา
3. จากนั้นก็ย้ายจานหมายเลข **n** จาก **left** ไป **right**
4. ตามด้วยการทำ **hanoi(n-1,mid,left,right)** คือย้ายจานหมายเลข 1 ถึง **n-1** จาก **mid** (ที่เป็นผลจากขั้นตอนที่ 2) ไป **right** (ซึ่งจะทับใบหมายเลข **n** ที่เราย้ายในขั้นตอนที่ 3)

ปัญหาข้างบนนี้มีชื่อว่า **Tower of Hanoi** แต่ปัญหาที่จะให้เขียนในโจทย์นี้เรียกว่า **Double Hanoi** เหมือน **Tower of Hanoi** บวกเงื่อนไขเพิ่มเติมคือ มีจาน **2n** ใบ จานแต่ละหมายเลขมี **2** ใบ (คือแต่ละขนาดมี **2** ใบ) แต่มีสีต่างกัน ใบหนึ่งดำ อีกใบหนึ่งขาว (จึงมีจานหมายเลข **1W,1B,2W,2B,...,nW,nB**)

- ตอนเริ่มต้น จานวางทับกันจากบนลงล่าง ใบบนไม่ใหญ่กว่าใบล่าง และวางสลับสีกัน
- ตอนย้ายจาน ก็เหมือนเดิม ย้ายได้ทีละใบ ใบใหญ่ห้ามทับใบเล็ก
- เป้าหมายคือ ย้ายจานทั้ง **2n** ใบ จากเสาซ้ายไปไว้ที่เสาขวา ที่วางในลักษณะเดียวกัน



รูปที่ 2 Double Hanoi

ขอเสนอวิธีการย้ายแบบง่าย ๆ (ที่ไม่ได้จำนวนครั้งการย้ายที่น้อยสุดนะ) ให้ทำตามรูปที่ 2 ทางขวานี้

1. เริ่มทำ **Double Hanoi** เพื่อย้ายจาน **1W,1B** ถึง **(n-1)W,(n-2)B** จากเสาซ้ายไปเสาขวา
2. ย้ายจานหมายเลข **nW** จากเสาซ้ายไปกลาง ตามด้วยย้ายจานหมายเลข **nB** จากเสาซ้ายไปกลาง
3. ทำ **Double Hanoi** เพื่อย้ายจาน **1W,1B** ถึง **(n-1)W,(n-2)B** จากเสาขวากลับมาเสาซ้าย
4. และ 5 น่าจะเดาได้จากรูปว่าต้องทำอะไร

จงปรับฟังก์ชันข้างล่างนี้ให้ทำตามขั้นตอนที่กำหนดไว้

```
def dhanoi(n,left,mid,right):
  if n == 0 : return
  dhanoi(n-1,_____________________________)     # ❶
  print(str(n)+'W',':',left,'->',_____________)  #   ❷
  print(str(n)+'B',':',left,'->',_____________)  #   ❷
  dhanoi(n-1,_____________________________)     # ❸
  ???                                           #   ❹
                                                #   ❹
                                                # ❺
exec(input().strip()  # don't remove this line
```

ข้อมูลนำเข้า

คำสั่งในการทดสอบฟังก์ชันที่เขียน

ข้อมูลส่งออก

ผลที่ได้จากคำสั่งที่ป้อนเป็นข้อมูลนำเข้า

ตัวอย่าง

input (ทางแป้นพิมพ์)	output (ทางจอภาพ)
`dhanoi(2,'L','M','R')`	1W : L -> M 1B : L -> M 1B : M -> R 1W : M -> R 2W : L -> M 2B : L -> M 1W : R -> M 1B : R -> M 1B : M -> L 1W : M -> L 2B : M -> R 2W : M -> R 1W : L -> M 1B : L -> M 1B : M -> R 1W : M -> R
`dhanoi(3,'L','M','R')`	1W : L -> M 1B : L -> M 1B : M -> R 1W : M -> R 2W : L -> M 2B : L -> M 1W : R -> M 1B : R -> M 1B : M -> L 1W : M -> L 2B : M -> R 2W : M -> R 1W : L -> M 1B : L -> M 1B : M -> R 1W : M -> R 3W : L -> M 3B : L -> M 1W : R -> M 1B : R -> M 1B : M -> L 1W : M -> L 2W : R -> M 2B : R -> M 1W : L -> M 1B : L -> M 1B : M -> R 1W : M -> R 2B : M -> L 2W : M -> L 1W : R -> M 1B : R -> M 1B : M -> L 1W : M -> L 3B : M -> R 3W : M -> R 1W : L -> M 1B : L -> M 1B : M -> R 1W : M -> R 2W : L -> M 2B : L -> M 1W : R -> M 1B : R -> M 1B : M -> L 1W : M -> L 2B : M -> R 2W : M -> R 1W : L -> M 1B : L -> M 1B : M -> R 1W : M -> R

R-81: เกมซูโดกุ

คงเคยเห็นหรือเล่นเกม **Sudoku** กันมาบ้าง ตัวอย่างเช่น จากรูปด้านซ้าย เราต้องเติมตัวเลขให้ครบทุกช่อง โดยไม่มีแถวแนวนอนใดมีเลขซ้ำกัน ไม่มีแถวแนวตั้งใดมีเลขซ้ำกัน และไม่มีเลขซ้ำกันในกลุ่ม **3x3** เดียวกัน

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

โจทย์ข้อนี้ให้เขียนโปรแกรมเพื่อหาคำตอบของกระดานเริ่มต้นที่ได้รับ (ศึกษารายละเอียดของฟังก์ชัน และจาก **comment**)

ฟังก์ชัน **solve(board)** รับตาราง **Sudoku** ที่ให้มาในรูปของสตริงของตัวเลข **81** ตัว เรียงจากซ้ายไปขวา จากบนลงล่าง ตำแหน่งที่ว่างแทนด้วยจุด เช่นตารางทางซ้ายข้างบนนี้ แทนด้วยสตริง

```
'53..7....6..195....98....6.8...6...34..8.3..17...2...6.6....28....419..5....8..79'
```

```
def same_row(i,j):          # i และ j คือ index ในสตริง 81 ตัวที่แทนตาราง 9x9
    return (i//9 == j//9)
def same_col(i,j):
    return (i-j) % 9 == 0
def same_block(i,j):
    return (i//27 == j//27 and i%9//3 == j%9//3)
def show(board):
    for i in range(3):
        print('+---+---+---+')
        for j in range(3):
            k = 9*(3*i+j)
            print('|'+board[k:k+3]+'|'+board[k+3:k+6]+'|'+board[k+6:k+9]+'|')
    print('+---+---+---+')

def solve(board):
    # หาซิว่า board ยังมี จุด เหลืออยู่ไหม ถ้าไม่มี ก็คืน board กลับไป
    ???
    # แต่ถ้ายังมี จุด อยู่ใน board
    # สร้างเซต S ซึ่งเก็บตัวเลขที่อยู่ในแถวแนวนอนเดียวกับจุด ตัวเลขที่อยู่ในแถวแนวตั้งเดียวกับจุด และตัวเลขที่อยู่ในกลุ่มเดียวกับจุด
    # (ทำง่าย ๆ ด้วยการลุยทุกตัวใน board แล้วใช้ฟังก์ชัน same_row, same_col, same_block ให้เป็นประโยชน์)
    ???

    # ให้ T = เซตของเลข '0' ถึง '9' ที่ไม่มีข้อมูลในเซต S (T ก็คือเซตที่เก็บเลขที่อาจใช้แทนจุดได้ )
    ???

    for e in T:
       newboard = ???               # สร้างกระดานใหม่มีค่าเหมือน board แต่แทนจุดด้วย e
       sol = solve(newboard)        # จำนวนจุดลดลงหนึ่ง ลองลุยหาคำตอบต่อ
       if sol != '' : return sol    # ถ้าลองแล้วใช้ได้ ก็คืนผล
    return ''                       # ถ้าลองทุกแบบ แล้วไม่สำเร็จ ก็คืน '' บอกว่าไม่พบคำตอบ

sol = solve(input().strip())
show(sol)
```

ข้อมูลนำเข้า

สตริงของเลข 0 ถึง 9 และจุด จำนวน 81 ตัว ที่แทนตารางเริ่มต้นของเกม Sudoku โดยจุดแทนช่องที่ยังไม่เติมเลข

ข้อมูลส่งออก

ผลลัพธ์ที่ได้จากการเติมเลข (เพื่อความง่าย input ที่ใช้ในการตรวจจะหาคำตอบได้เสมอ และมีเพียงคำตอบเดียวแน่ ๆ)

ตัวอย่าง

input (ทางแป้นพิมพ์)

```
.2.......6....3.74.8.........3..2.8..4..1.6..5.........1.78.5....9.........4.
```

	2							
			6					3
	7	4		8				
					3			2
	8			4			1	
6			5					
				1		7	8	
5					9			
							4	

output (ทางจอภาพ)

```
+---+---+---+
|126|437|958|
|895|621|473|
|374|985|126|
+---+---+---+
|457|193|862|
|983|246|517|
|612|578|394|
+---+---+---+
|269|314|785|
|548|769|231|
|731|852|649|
+---+---+---+
```

1	2	6	4	3	7	9	5	8
8	9	5	6	2	1	4	7	3
3	7	4	9	8	5	1	2	6
4	5	7	1	9	3	8	6	2
9	8	3	2	4	6	5	1	7
6	1	2	5	7	8	3	9	4
2	6	9	3	1	4	7	8	5
5	4	8	7	6	9	2	3	1
7	3	1	8	5	2	6	4	9

คำเตือน : ตัวอย่างข้างบนนี้เป็นโจทย์ Sudoku ที่ไม่ง่าย ใช้เวลาหาคำตอบพอสมควร

R-82: ถอดความ

คุณต้องการจะส่งข้อความหนึ่งให้กับเพื่อนของคุณ แต่ไม่อยากจะให้คนอื่นรู้ จึงได้เติมตัวอักษรหลอกลงไปในข้อความเพื่อให้ตีความยากขึ้น ตัวอักษรในข้อความจะเป็นตัวพิมพ์เล็กหรือพิมพ์ใหญ่ก็ได้ แต่ตัวอักษรหลอกที่เพิ่มเข้าไป จะเป็นตัวพิมพ์ใหญ่เสมอ

ตัวอย่างเช่น ข้อความเริ่มต้นคือ **HelloWorld** เมื่อเติมตัวอักษรหลอกแล้วอาจจะได้เป็น **HYelloWorMld** เป็นต้น เพื่อนของคุณจึงต้องลองลบอักษรตัวพิมพ์ใหญ่ทุกแบบ เพื่อดูว่าข้อความจริงเป็นอย่างไร

จงเขียนโปรแกรมรับข้อความที่เติมอักษรหลอกแล้ว จากนั้นแสดงการลบอักษรพิมพ์ใหญ่ทุกแบบที่เป็นไปได้

ข้อมูลนำเข้า

ข้อความภาษาอังกฤษ 1 บรรทัด ประกอบด้วยตัวอักษรภาษาอังกฤษพิมพ์ใหญ่และพิมพ์เล็ก

ข้อมูลส่งออก

แสดงผลของการลบอักษรพิมพ์ใหญ่ทุกแบบที่เป็นไปได้จากข้อความที่ได้รับ โดยเรียงลำดับตามพจนานุกรม ถ้ามีข้อความที่ซ้ำกันให้แสดงครั้งเดียว

ตัวอย่าง

input (ทางแป้นพิมพ์)	output (ทางจอภาพ)
HYelloWorMld	**HYelloWorMld** **HYelloWorld** **HYelloorMld** **HYelloorld** **HelloWorMld** **HelloWorld** **HelloorMld** **Helloorld** **YelloWorMld** **YelloWorld** **YelloorMld** **Yelloorld** **elloWorMld** **elloWorld** **elloorMld** **elloorld**
PPAP	**A** **AP** **P** **PA** **PAP** **PP** **PPA** **PPAP** **PPP**
letter	**letter**
Zebra	**Zebra** **ebra**

R-83: การเดินทางผ่านจุดวาร์ป

อาคารแห่งหนึ่งมีห้องมากมาย แต่ละห้องมีหมายเลขเป็นจำนวนเต็ม 1, 2, 3, ... ไม่สิ้นสุด การที่จะเดินทางจากห้องหนึ่งไปอีกห้องนั้นต้องใช้จุดวาร์ป จุดวาร์ปจะมีอยู่ในบางห้องเท่านั้น การใช้จุดวาร์ปจะทำให้สามารถเดินทางจากห้อง x ไปยังห้อง y ได้ (x -> y) และจุดวาร์ปจะพาไปห้องที่มีหมายเลขเพิ่มขึ้นเสมอ เช่น อาจจะมีจุดวาร์ปจากห้องที่ 7 ไปห้องที่ 13 (7 -> 13) แต่จะไม่มีทางมีจุดวาร์ปจากห้องที่ 10 ไปห้องที่ 8 (10 -> 8) และก็ไม่มีจุดวาร์ปเข้าออกห้องเดียวกัน

ปัญหาคือ ขณะนี้คุณอยู่ที่ห้องที่ a และคุณอยากไปกินขนมแสนอร่อยที่อยู่ที่ห้อง b ให้หาว่า มีวิธีการใช้จุดวาร์ปเพื่อเดินทางจากห้อง a ไปยังห้อง b หรือไม่ วิธีการเดินทางจะผ่านจุดวาร์ปกี่ครั้งก็ได้เช่น เดินทางจากห้อง 3 ไปห้อง 20 อาจเป็น 3 -> 5 -> 14 -> 20 เป็นต้น

ข้อมูลนำเข้า

บรรทัดแรกมีเลข 3 ตัว เป็นจำนวนจุดวาร์ปทั้งหมดในอาคาร หมายเลขห้องปัจจุบัน (a) และหมายเลขห้องของขนมแสนอร่อย (b) จากนั้นบรรทัดที่เหลือจะเป็นข้อมูลของจุดวาร์ป โดยแต่ละบรรทัดจะมีเลข 2 ตัว คือ x และ y บอกว่ามีจุดวาร์ปจากห้องที่ x ไปห้องที่ y *** รับประกันว่า a < b และ x < y สำหรับทุกจุดวาร์ป ***

ข้อมูลส่งออก

มีบรรทัดเดียว ถ้ามีวิธีการใช้จุดวาร์ปในการเดินทางจากห้อง a ไปยังห้อง b ให้พิมพ์ว่า **yes**
ถ้าไม่มีให้พิมพ์ว่า **no** (ตัวพิมพ์เล็ก)

ตัวอย่าง

input (ทางแป้นพิมพ์)	output (ทางจอภาพ)
5 3 20 3 5 5 14 14 20 5 18 4 20	yes
5 3 20 3 21 3 6 6 7 7 19 6 19	no
5 2 10 3 10 1 7 2 3 2 10 7 10	yes
1 1 100 3 7	no

R-84: หาวิธีการเดินทางผ่านจุดวาร์ป

อาคารแห่งหนึ่งมีห้องมากมาย แต่ละห้องมีหมายเลขเป็นจำนวนเต็ม 1, 2, 3, ... ไม่สิ้นสุด การที่จะเดินทางจากห้องหนึ่งไปอีกห้องนั้นต้องใช้จุดวาร์ป จุดวาร์ปจะมีอยู่ในบางห้องเท่านั้น การใช้จุดวาร์ปจะทำให้สามารถเดินทางจากห้อง x ไปยังห้อง y ได้ (x -> y) และจุดวาร์ปจะพาไปห้องที่มีหมายเลขเพิ่มขึ้นเสมอ เช่น อาจจะมีจุดวาร์ปจากห้องที่ 7 ไปห้องที่ 13 (7 -> 13) แต่จะไม่มีทางมีจุดวาร์ปจากห้องที่ 10 ไปห้องที่ 8 (10 -> 8) และก็ไม่มีจุดวาร์ปเข้าออกห้องเดียวกัน

ปัญหาคือ ขณะนี้คุณอยู่ที่ห้องที่ a และคุณอยากไปกินขนมแสนอร่อยที่อยู่ที่ห้อง b ให้หาวิธีการใช้จุดวาร์ปเพื่อเดินทางจากห้อง a ไปยังห้อง b วิธีการเดินทางจะผ่านจุดวาร์ปกี่ครั้งก็ได้เช่น เดินทางจากห้อง 3 ไปห้อง 20 อาจเป็น 3 -> 5 -> 14 -> 20 เป็นต้น

ข้อมูลนำเข้า

บรรทัดแรกมีเลข 3 ตัว เป็นจำนวนจุดวาร์ปทั้งหมดในอาคาร หมายเลขห้องปัจจุบัน (a) และหมายเลขห้องของขนมแสนอร่อย (b) จากนั้นบรรทัดที่เหลือจะเป็นข้อมูลของจุดวาร์ป โดยแต่ละบรรทัดจะมีเลข 2 ตัว คือ x และ y บอกว่ามีจุดวาร์ปจากห้องที่ x ไปห้องที่ y *** รับประกันว่า a < b และ x < y สำหรับทุกจุดวาร์ป ***

ข้อมูลส่งออก

ถ้ามีวิธีการใช้จุดวาร์ปในการเดินทางจากห้อง a ไปยังห้อง b ให้แสดงทุกวิธีการเดินทางทั้งหมดทีละบรรทัด โดยแสดงห้องหมายเลขน้อยก่อน สำหรับแต่ละวิธีการเดินทาง ให้แสดงหมายเลขห้องคั่นด้วย **->** (ดูตัวอย่าง) ถ้าไม่มีวิธีการเดินทางให้แสดง **no**

ตัวอย่าง

input (ทางแป้นพิมพ์)	output (ทางจอภาพ)
5 3 20 3 5 5 14 14 20 5 18 4 20	3 -> 5 -> 14 -> 20
5 3 20 3 21 3 6 6 7 7 19 6 19	no
5 2 10 3 10 1 7 2 3 2 10 7 10	2 -> 3 -> 10 2 -> 10
1 1 100 3 7	no

R-85: แปลงรูปแบบวิถี

มีเมืองหลายเมือง แต่ละเมืองแทนด้วยเลขจำนวนเต็ม สมมติว่า เรามีทางเดินสองทางที่เป็นลำดับของเลขเมืองดังนี้

`3, 5, 8` และ `3, 5, 6, 7`

ถ้าเราเก็บทางเดินเหล่านี้ด้วยลิสต์เป็น `[ [3,5,8], [3,5,6,7] ]` จะเห็นมีทางเดินส่วนหน้าซ้ำกัน สามารถเก็บแบบทางเดินร่วมแยกออกมาเป็น `[3, [5, [[8], [6, [7]]]]]` ซึ่งขออธิบายแนวทางเก็บแบบหลังนี้ด้วยตัวอย่างในตารางข้างล่างนี้

ลิสต์ที่เก็บทางเดินแบบ **A**	ลิสต์ที่เก็บทางเดินแบบ **B**
`[ [3] ]`	`[ 3 ]`
`[ [3,4] ]`	`[ 3,[4] ]`
`[ [3,4,5] ]`	`[ 3,[4,[5]] ]`
`[ [7], [8,9] ]`	`[ [7],[8,[9]] ]`
`[ [3,7], [3,8,9] ]`	`[ 3,[[7],[8,[9]]] ]`
`[ [3,7], [3,8,9], [2,4,5] ]`	`[ [3,[[7],[8,[9]]]], [2,[4,[5]]] ]`
`[ [1,3,7], [1,3,8,9], [1,2,4,5] ]`	`[ 1, [ [3,[[7],[8,[9]]]], [2,[4,[5]]] ] ]`

จงเขียนฟังก์ชัน `B_to_A` ที่รับลิสต์ทางเดินแบบ `B` แล้วแปลงให้เป็นลิสต์ทางเดินแบบ `A`

```
def B_to_A( b ):
 ???

print(B_to_A(eval(input().strip()))) # do not remove this line
```

ข้อมูลนำเข้า

ลิสต์แบบ B

ข้อมูลส่งออก

ลิสต์แบบ A

ตัวอย่าง

input (ทางแป้นพิมพ์)	output (ทางจอภาพ)
`[ 3 ]`	`[[3]]`
`[ 3,[4] ]`	`[[3, 4]]`
`[ 3,[4,[5]] ]`	`[[3, 4, 5]]`
`[ [7],[8,[9]] ]`	`[[7], [8, 9]]`
`[ 3,[[7],[8,[9]]] ]`	`[[3, 7], [3, 8, 9]]`
`[ [3,[[7],[8,[9]]]], [2,[4,[5]]] ]`	`[[3, 7], [3, 8, 9], [2, 4, 5]]`
`[ 1, [ [3,[[7],[8,[9]]]], [2,[4,[5]]] ] ]`	`[[1, 3, 7], [1, 3, 8, 9], [1, 2, 4, 5]]`

R-86: การหาคำในตาราง

การหาคำในตาราง (Word Search Puzzle) ของโจทย์นี้มีข้อพิเศษตรงที่ ให้หาคำที่เรียงติดกันไม่ใช่แค่ตามแนวนอน แนวตั้ง แนวทแยง แต่ต้องหาคำที่ติดกันแบบยีกๆ ยือๆ ได้ด้วย (ตัวอักษรติดกันในที่นี้คือ ตัวอักษรที่ $k+1$ อยู่ด้านบน ซ้าย ล่าง หรือขวา ของตัวอักษรตัวที่ k) เช่น คำว่า **DRAGON** ปรากฏในตัวอักษรตัวสีแดงที่ติดกันดังแสดงในตารางทางขวานี้

W (0,0)	C (0,1)	O (0,2)	N (0,3)	S (0,4)
U (1,0)	D (1,1)	G (1,2)	Y (1,3)	L (1,4)
Z (2,0)	R (2,1)	A (2,2)	P (2,3)	O (2,4)
R (3,0)	T (3,1)	E (3,2)	K (3,3)	S (3,4)
I (3,0)	X (4,1)	F (4,2)	J (4,3)	E (4,4)

จงเขียนโปรแกรมที่รับตารางของตัวอักษร และคำที่ต้องการหา โดยแสดงผลลัพธ์เป็น list of tuple coordinates

ข้อมูลนำเข้า

- บรรทัดแรกระบุจำนวนแถว n และจำนวนคอลัมน์ m คั่นด้วยเว้นวรรค
- บรรทัดที่สอง ถึงบรรทัดที่ $n+1$ ระบุตารางอักษร G บรรทัดละแถว ประกอบด้วยตัวภาษาอังกฤษพิมพ์ใหญ่ m ตัว อยู่ติดกันหมด ไม่มีเว้นวรรค
- บรรทัดที่ $n+2$ คือ *word* ซึ่งเป็นคำที่ต้องการให้หา เป็นตัวพิมพ์ใหญ่ทั้งหมดเช่นเดียวกัน

ข้อมูลส่งออก

แสดงผลเป็น list of tuples ซึ่ง list นี้มีความยาวเท่ากับความยาวของ *word* แต่ละ tuple เก็บค่าพิกัดของตัวอักษร โดยเก็บแบบ (row_index, column_index)

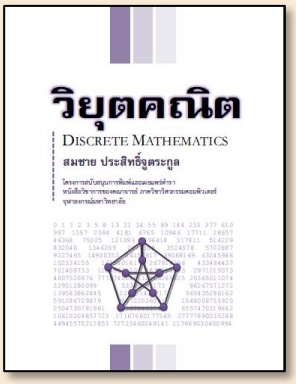
** หมายเหตุ รับประกันว่าตารางอักษร G จะมี *word* ที่หาได้แน่ ๆ และจะมีเพียงคำตอบเดียวเท่านั้น

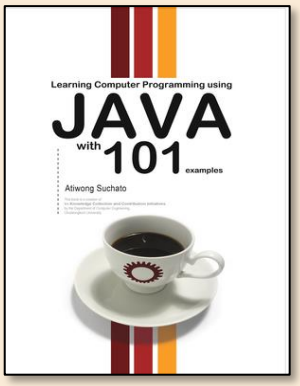
ตัวอย่าง

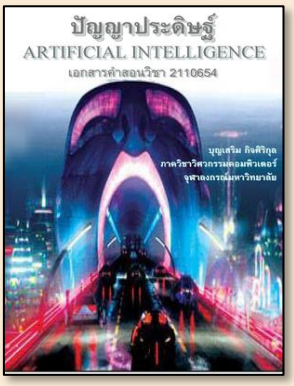
input (จากแป้นพิมพ์)	output (ทางจอภาพ)
3 4 DAYP DSTZ NOHJ PYTHON	[(0, 3), (0, 2), (1, 2), (2, 2), (2, 1), (2, 0)]
4 6 UHRVTG LCDKQH AILEUJ RKWSRH CHULA	[(1, 1), (0, 1), (0, 0), (1, 0), (2, 0)]
5 7 HCVXYYL NWOXYEC OPROANG SXREIGC JYJENDD ENGINEER	[(1, 5), (2, 5), (3, 5), (3, 4), (4, 4), (4, 3), (3, 3), (3, 2)]

หนังสือเล่มอื่นที่อาจสนใจ

กดที่หน้าปกหนังสือเพื่อดูรายละเอียด







https://www.cp.eng.chula.ac.th/books

แบบฝึกปฏิบัติ
เริ่มเรียนเขียนโปรแกรม

แบบฝึกปฏิบัติเริ่มเรียนเขียนโปรแกรมเล่มนี้รวบรวมโจทย์ปัญหาน่าสนใจในหลายระดับความยากง่ายที่นิสิตต้องฝึกปฏิบัติโดยใช้ระบบตรวจโปรแกรมอัตโนมัติ Grader ในการตรวจสอบความถูกต้องของโปรแกรม พร้อมกับลิงค์ไปยังวิดีทัศน์ทบทวนเนื้อหาและเฉลย ให้นิสิตเข้าใจหลักการในการใช้คำสั่งของภาษาโปรแกรม เพื่อเขียนโปรแกรมคอมพิวเตอร์ให้ตรงตามข้อกำหนดที่ได้รับ นิสิตพึงตระหนักว่า การเขียนโปรแกรมเป็นความสามารถที่พัฒนาได้มาด้วยการลงมือปฏิบัติด้วยตนเอง เหมือนกับทักษะอื่นทางวิศวกรรมที่ไม่สามารถได้มาด้วยการอ่าน ๆ ๆ แต่จำเป็นต้องฝึก ๆ ๆ เท่านั้น