บาร์โค้ด

บาร์โค้ด (Barcode หรือ รหัสแท่ง) คือ ภาพเครื่องหมายแทนข้อมูลที่สามารถใช้เครื่องอ่านเพื่อนำข้อมูลไปประมวลผลได้ง่าย เร็ว ถูกต้อง และแม่นยำ บาร์โค้ดประกอบด้วยเส้นมืด (มักจะเป็นสีดำ) และเส้นสว่าง (มักเป็นสีขาว) วางเรียงกันเป็นแนวดิ่ง แทนรหัสของตัวเลขและตัวอักษร เครื่องอ่านบาร์โค้ด (Barcode Scanner) อาศัยแสงยิงกวาดกระทบบาร์โค้ดในลักษณะพาดขวาง แสงที่สะท้อนจากเส้นมืดจะน้อยกว่าแสงที่สะท้อนจากพื้นที่สว่าง จึงสามารถแยกความกว้างระหว่างพื้นที่มืดและพื้นที่สว่างออกมาเป็นรหัส บาร์โค้ดจึงช่วยลดความผิดพลาดในการป้อนข้อมูลได้มาก

EAN-13 (European Article Numbering) เป็นประเภทบาร์โค้ดที่ใช้อย่างกว้างขวาง นิยมใช้กับสินค้าอุปโภคบริโภคทั่วไป เช่น อาหาร เครื่องดื่ม บาร์โค้ด EAN-13 ประกอบด้วยตัวเลข 13 หลัก มีความหมายดังนี้

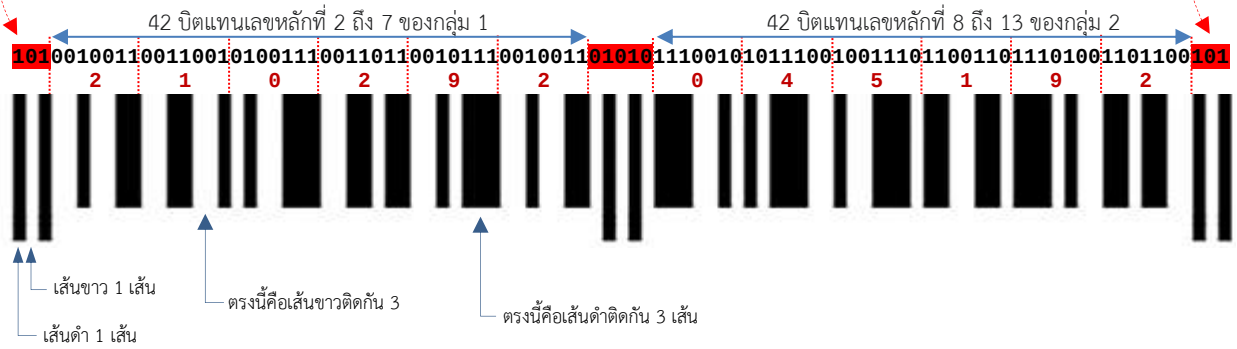
- 3 หลักแรก คือ รหัสของประเทศผู้ผลิต
- 4 หลักถัดมา คือ รหัสโรงงานที่ผลิต
- 5 หลักถัดมา คือ รหัสของสินค้า
- และตัวเลขในหลักสุดท้าย จะเป็นตัวเลขตรวจสอบความถูกต้องของบาร์โค้ด (check digit)

(ที่มา: http://www.barcodethai.com/?page_id=2155)

โครงสร้างของบาร์โค้ด EAN-13

บาร์โค้ด (รูปข้างล่างเป็นบาร์โค้ดของหมายเลข **3210292045192** ที่ขยายให้เห็นชัด ๆ) ประกอบด้วยเส้นตรงจำนวน 95 เส้นความกว้างเท่า ๆ กัน เส้นสีขาวแทนรหัส 0 เส้นสีดำแทนรหัส 1 ขอเรียกแต่ละเส้นว่า *บิต* บาร์โค้ดทั้ง 95 บิตนี้ประกอบกันเพื่อเข้ารหัสเลข 13 หลัก (ให้เลขหลักที่ 1 คือหลักซ้ายสุด เลขหลักที่ 13 คือหลักขวาสุด) บิตทั้ง 95 บิต (เรียงจากซ้ายไปขวา) จัดแบ่งเป็นส่วน ๆ ได้ดังนี้

- 3 บิตแรกมีค่า 101 เสมอ เป็นค่าบอกการเริ่มต้นของบาร์โค้ด
- 42 บิตถัดมาแทนเลขหลักที่ 2 ถึง 7 (6 หลัก หลักละ 7 บิต) เรียกว่า *กลุ่ม 1*
- 5 บิตถัดมามีค่า 01010 เสมอ เป็นค่าแบ่งตัวเลขเป็นสองกลุ่ม
- 42 บิตถัดมาแทนเลขหลักที่ 8 ถึง 13 (6 หลัก หลักละ 7 บิต) เรียกว่า *กลุ่ม 2*
- 3 บิตสุดท้ายมีค่า 101 เสมอ เป็นค่าบอกการสิ้นสุดของของบาร์โค้ด
- ข้อสังเกต: อาจสงสัยว่า
  - เลขหลักที่ 1 เก็บอยู่ที่ใด ? เลขหลักที่ 1 มีค่าที่ถอดได้จากบิตต่าง ๆ ในกลุ่ม 1 ที่จะได้กล่าวต่อไป
  - เนื่องจากเริ่มและจบด้วย 101 และรหัสแบ่งกลางก็เป็น 01010 ดังนั้น หากเราวางบาร์โค้ดกลับหัว แล้วสแกนรหัสเข้ามา จะรู้ได้อย่างไร ? ตรงนี้จะแยกด้วยรูปแบบบิตของกลุ่ม 1 และกลุ่มที่ 2 ที่ต่างกัน ที่จะได้กล่าวต่อไป



รูปแบบของบิตที่แทนเลขในบาร์โค้ด

ขออธิบายเลขในกลุ่ม 1 (เลขหลักที่ 2 ถึง 7) และกลุ่ม 2 (เลขหลักที่ 8 ถึง 13) ก่อน เลขแต่ละหลักถูกเข้าเป็นรหัส 0/1 ขนาด 7 บิตในรูปแบบดังนี้

- เลขในกลุ่ม 1 ถูกเข้ารหัสให้บิตซ้ายสุดเป็น 0 เสมอ แบ่งย่อยเป็น
  - รูปแบบ L เป็นรูปแบบที่มีบิต 1 เป็นจำนวนคี่ เช่น `0001101` (มีบิต 1 สามบิต)
  - รูปแบบ G เป็นรูปแบบที่มีบิต 1 เป็นจำนวนคู่ เช่น `0111001` (มีบิต 1 สี่บิต)
- เลขในกลุ่ม 2 ถูกเข้ารหัสให้บิตซ้ายสุดเป็น 1 เสมอ เรียกว่ารูปแบบ R เช่น `1011100`

ตารางทางขวากำหนดรหัสของเลข 0 ถึง 9 ในรูปแบบ L, G และ R ตามมาตรฐาน EAN-13

เลข	รหัสของเลข		
	รูปแบบ L	รูปแบบ G	รูปแบบ R
0	0001101	0100111	1110010
1	0011001	0110011	1100110
2	0010011	0011011	1101100
3	0111101	0100001	1000010
4	0100011	0011101	1011100
5	0110001	0111001	1001110
6	0101111	0000101	1010000
7	0111011	0010001	1000100
8	0110111	0001001	1001000
9	0001011	0010111	1110100

ตัวอย่าง ถ้าเลขในกลุ่ม 2 ทั้ง 6 หลักคือ **045192** จะถูกเข้ารหัส (ด้วยรูปแบบ R ทั้งหมด) เป็น

```
1110010 1011100 1001110 1100110 1110100 1101100
   0       4       5       1       9       2
```

รูปแบบของเลขในกลุ่ม 1

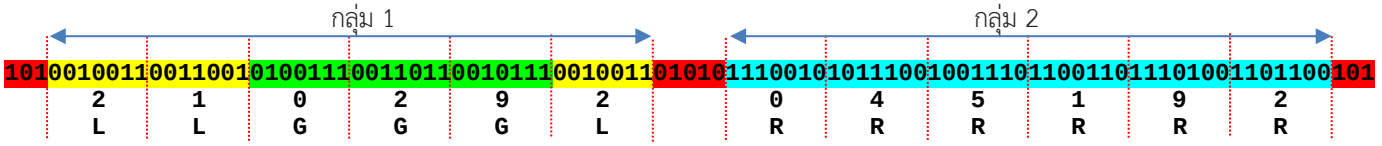
แล้วเลขในกลุ่ม 1 จะถูกเข้ารหัสอย่างไร ? มาตรฐาน EAN-13 กำหนดว่า เลขแต่ละหลักในกลุ่ม 1 ถูกเข้ารหัสด้วยรูปแบบ L หรือ G ขึ้นกับว่า **เลขหลักที่ 1** มีค่าเท่าใด ซึ่งดูได้จากตารางทางขวานี้

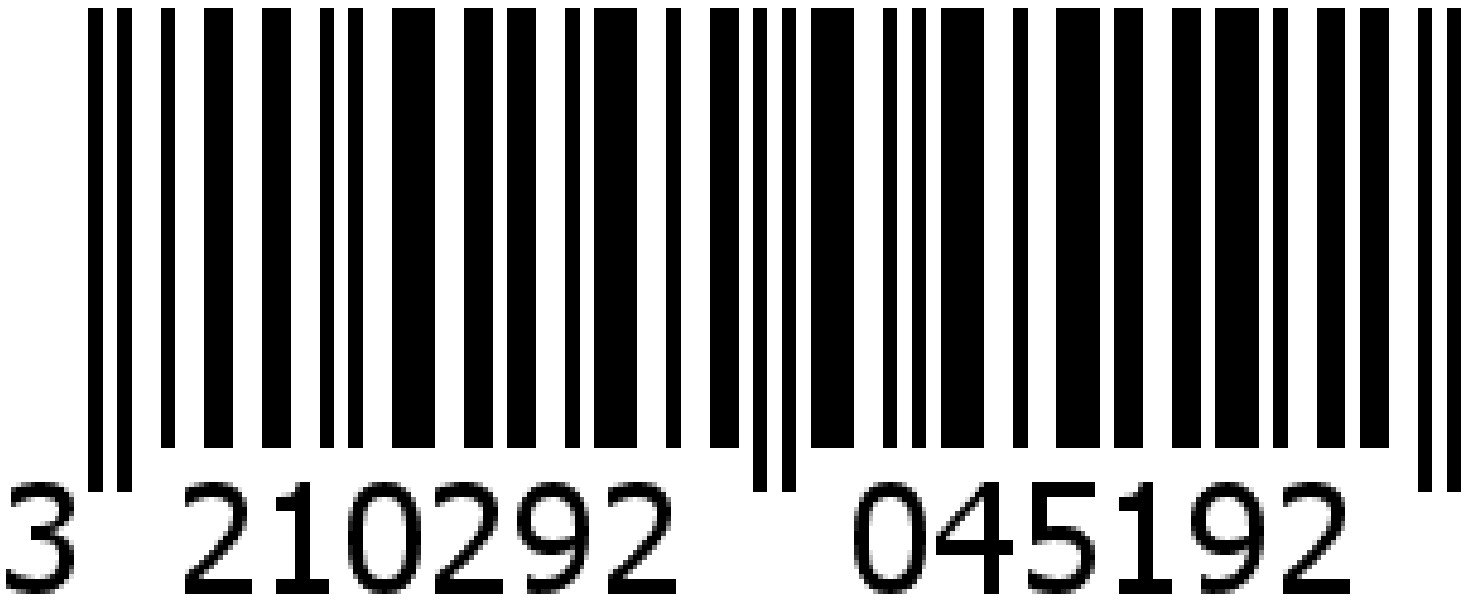
เลขหลักที่ 1	รูปแบบการเข้ารหัสของเลขหลักที่ 2 ถึง 7
0	LLLLLL
1	LLGLGG
2	LLGGLG
3	LLGGGL
4	LGLLGG
5	LGGLLG
6	LGGGLL
7	LGLGLG
8	LGLGGL
9	LGGLGL

ตัวอย่าง ถ้าเลขหลักที่ 1 ถึง 7 คือ **3210292** เนื่องจากเลขหลักที่ 1 คือ **3** จากตารางจะได้ว่าเลขหลักที่ 2 ถึง 7 ของกลุ่ม 1 ต้องถูกเข้ารหัสด้วยรูปแบบ **LLGGGL** ตามลำดับ (คือเลขหลักที่ 2, 3 และ 7 ถูกเข้ารหัสในรูปแบบ L และเลขหลักที่ 4,5 และ 6 ถูกเข้ารหัสในรูปแบบ G ได้ผลลัพธ์เป็น

```
0010011 0011001 0100111 0011011 0010111 0010011
   2       1       0       2       9       2
   L       L       G       G       G       L
```

ดังนั้น หากเลขทั้ง 13 หลักคือ **3210292045192** จะแปลงเป็นบิตทั้งหมด 95 บิต ดังรูปข้างล่างนี้





เลขตรวจสอบ (Check Digit)

ตามที่กล่าวไว้ตอนต้นว่า ตามมาตรฐาน EAN-13 เลขหลักที่ 13 คือ check digit หมายความว่า เป็นเลขนี้ที่มีไว้ตรวจสอบความถูกต้องของ 12 หลักแรก เลขหลักที่ 13 ได้มาจากการนำเลขหลักที่ 1 ถึง 12 มาคำนวนตามขั้นตอนที่จะอธิบายต่อไป เมื่อเครื่องอ่านบาร์โค้ดถอดรหัสเลขต่าง ๆ ได้แล้ว ระบบต้องตรวจว่า เลขหลักที่ 13 ที่อ่านมาได้ กับค่า check digit ที่คำนวณจาก 12 หลักแรกมีค่าเท่ากันหรือไม่ ถ้าเท่า ก็แสดงว่าเลขที่อ่านมาน่าจะมีความถูกต้องสูง แต่ถ้าไม่เท่า ก็ส่งสัญญาณให้ผู้ใช้สแกนบาร์โค้ดใหม่ วิธีหาค่า check digit จากเลข 12 หลักแรก ทำได้ดังนี้

1. นำค่าเลขหลักต่าง ๆ 12 หลักแรก เขียนเรียงจากซ้ายไปขวา
2. เขียนเลข 1 กับ 3 สลับกันไป จากซ้ายไปขวาให้ตรงกับเลข 12 หลักข้างบน
3. หาผลรวมของผลคูณของเลขในข้อ 1 และ 2 ที่ตำแหน่งตรงกัน
4. หาเลขที่เป็นจำนวนเท่าของ 10 ที่มีค่าน้อยสุด ที่ไม่ต่ำกว่าค่าในข้อ 3
5. check digit มีค่าเท่ากับผลต่างของค่าในข้อ 4 กับ 3

ตัวอย่าง: เลข 12 หลัก คือ 321029204519											
3	2	1	0	2	9	2	0	4	5	1	9
1	3	1	3	1	3	1	3	1	3	1	3
3 + 6 + 1 + 0 + 2 +27 + 2 + 0 + 4 +15 + 1 +27 = 88											
90											
check digit = 90 − 88 = **2**											

สิ่งที่ต้องทำ

ก่อนจะอ่านว่าต้องเขียนฟังก์ชันอะไร ขออธิบายความหมายของศัพท์ที่ใช้ในการตั้งชื่อฟังก์ชันและพารามิเตอร์ ดังนี้

- `code` คือ รหัสของเลข `0` กับ `1` เช่น `code` แบบ `L` ของ `'4'` คือ `'0100011'`
- `digit` คือ เลข `0` ถึง `9`
- `pattern` คือ รูปแบบ `L`, `G` หรือ `R`
- `codes` คือ หลาย `code` ต่อ ๆ กัน
- `digits` คือ หลาย `digit` ต่อ ๆ กัน
- `patterns` คือ หลาย `pattern` ต่อ ๆ กัน

ให้สังเกตว่า ข้อมูลทั้งหลายที่ส่งไปให้ฟังก์ชัน หรือที่คืนมาจากฟังก์ชัน จะเป็นสตริง (ถึงแม้เราจะจัดการกับเลขก็ตาม แต่เป็นสตริงที่เก็บเลขโดด)

ฟังก์ชันที่ต้องเขียนให้สมบูรณ์มีดังนี้

```
def codes_of( digits, patterns ):
    # คืน codes ของเลขแต่ละตัวใน digits ที่เข้ารหัสตามรูปแบบที่กำหนดใน patterns ตามลำดับ
    # หมายเหตุ: digits และ patterns มีความยาวเท่ากันแน่ ๆ
    # เช่น codes_of('45', 'LG') คืน '01000110111001'
    #     code ของ '4' ในรูปแบบ 'L' คือ '0100011' และ code ของ '5' ในรูปแบบ 'G' คือ '0111001'
def digits_of( codes ):
    # คืน digits ของรหัสต่าง ๆ ใน codes
    # ถ้า codes ที่ได้รับมีบางรหัสภายในแปลงไม่ได้ ให้คืนสตริงว่าง
    # เช่น digits_of('01000110111001') คืน '45',
    #     digits_of('01000111111111') คืน ''
def patterns_of( codes ):
    # คืน patterns ของรหัสต่าง ๆ ใน codes
    # ถ้า codes ที่ได้รับมีรหัสที่ไม่อยู่ในรูปแบบใด ให้คืนสตริงว่าง
    # เช่น patterns_of('01000110111001') คืน 'LG'
    #     patterns_of('01000111111111') คืน ''
```

```
def check_digit( digits ):
    # คืน check digit ที่คำนวณจากเลข 12 หลักที่ได้รับใน digits ตามมาตรฐานของ EAN-13
    # หมายเหตุ: digits เป็นสตริงที่มี 12 หลักแน่ ๆ
    # เช่น  check_digit('321029204519') คืน '2'
    #       check_digit('123456789018') คืน '0'
```

```
def encode_EAN13( digits ):
    # คืนสตริงของ 0 และ 1 ความยาว 95 ตัว ที่ได้มาจากการเข้ารหัสเลขใน digits แบบ EAN-13
    # ถ้าเกิดกรณีต่อไปนี้ ให้คืนสตริงว่าง
    #   พบตัวที่ไม่ใช่ตัวเลขใน digits
    #   digits มีไม่ครบ 13 หลัก
    #   check digit ใน digits ไม่ตรงกับที่ควรจะเป็น
    # เช่น
    #   encode_EAN13('1234') คืน ''
    #   encode_EAN13('one-two-33333') คืน ''
    #   encode_EAN13('1111111111111') คืน ''
    #   encode_EAN13('3210292045192') คืน
    #   '10100100110011001010011100110110010111001001101010111001010111001001110110011011101001101100101'
```

```
def decode_EAN13( codes ):
    # คืนสตริงของเลขที่ได้จากการถอดรหัสจากสตริง 0/1 ที่เก็บใน codes แบบ EAN-13
    # ถ้าเกิดกรณีต่อไปนี้ ให้คืนสตริงว่าง (สาเหตุเหล่านี้มักมาจากเครื่องอ่านบาร์โค้ดอ่านรหัส 0 และ 1 มาผิด)
    #   codes เก็บจำนวนบิต หรือรูปแบบไม่ตรงข้อกำหนด
    #   รหัสบางส่วนแปลงเป็นตัวเลขไม่ได้
    #   เลขหลักที่ 13 ที่อ่านได้ ไม่ตรงกับค่า check digit ที่คำนวณได้
    # หมายเหตุ: เป็นไปได้ว่า ผู้ใช้เครื่องบาร์โค้ด อาจสแกนบาร์โค้ดที่วางกลับหัวก็ได้ ฟังก์ชันนี้ก็ต้องรองรับกรณีเช่นนี้ด้วย
    # เช่น
    #   c = '10100100110011001010011100110110010111001001101010111001010111001001110110011011101001101100101'
    #   decode_EAN13(c) คืน '3210292045192'
    #   c = '10111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111101'
    #   decode_EAN13(c) คืน ''
```

โปรแกรมต้นฉบับ

```
# Prog-07: EAN-13 Barcode
# ??3??????1 Name ?
```

ใส่เลขประจำตัว ชื่อ นามสกุล

download code นี้ได้

```
import math
import matplotlib.pyplot as plt
#-----------------------------------------------
def show_barcode(digits, ean13_code):
    x = [[int(e) for e in ean13_code]]
    plt.axis('off')
    plt.imshow(x, aspect='auto', cmap='binary')
    plt.title(digits)
    plt.show()
#-----------------------------------------------
def test1():
    digits = input('Enter a 13-digit number: ')
    codes = encode_EAN13(digits)
    if codes == '':
        print(digits, 'is not an EAN-13 number.')
    else:
        decoded_digits = decode_EAN13(codes)
        if decoded_digits == digits:
            show_barcode(digits, codes)
        else:
            print('Error in decoding.')
#-----------------------------------------------
L_codes = ['0001101', '0011001', '0010011', '0111101', '0100011', \
           '0110001', '0101111', '0111011', '0110111', '0001011']
G_codes = ['0100111', '0110011', '0011011', '0100001', '0011101', \
           '0111001', '0000101', '0010001', '0001001', '0010111']
R_codes = ['1110010', '1100110', '1101100', '1000010', '1011100', \
           '1001110', '1010000', '1000100', '1001000', '1110100']

#===============================================
```

โปรแกรมที่ส่ง ห้ามเปลี่ยนอะไรใด ๆ ในส่วนที่มีพื้นหลังเป็นสีแดง และที่เป็นตัวสีแดงโดยเด็ดขาด เปลี่ยนได้เฉพาะส่วนที่มีพื้นหลังเป็นสีเขียวเท่านั้น

```

```

บริเวณนี้ไว้เขียนข้อมูลหรือฟังก์ชันเสริมเพิ่มเอง (ถ้าต้องการ)

```
def codes_of(digits, patterns):
    
def digits_of(codes):
    
def patterns_of(codes):
    
def check_digit(digits):
    
def encode_EAN13(digits):
    
def decode_EAN13(codes):
    
#-----------------------------------------------
test1()
```

ห้าม import อะไรเพิ่มเติม และห้ามใช้ที่เก็บข้อมูลจำพวก set หรือ dict หรือที่คล้ายคลึงโดยเด็ดขาด

ควรใช้ฟังก์ชันที่ได้เขียนข้างบนให้เป็นประโยชน์ให้มากที่สุด