Compiler. All questions have the same score.

1) Change this regular expression into a railroad diagram

F(A|B|C)X\*Z

2) Left recursion in a grammar causes it to be non-LL(1). For example, this grammar has left recursion

A -> X Y | X Z

it can be "left factored" to change it to LL(1). The above grammar can be left-factored as:

A -> X A'

A' -> Y | Z

Now please do left-factored of the following grammar

ifst -> if ( exp ) st else st | if ( exp ) st

3) Compute the follow set of the following grammar

exp -> term exp’

exp’ -> addop term exp’ | e

addop -> + | -

term -> factor term’

term’ -> mulop factor term’ | e

mulop -> \*

factor -> ( exp ) | num

where e is an empty string

END