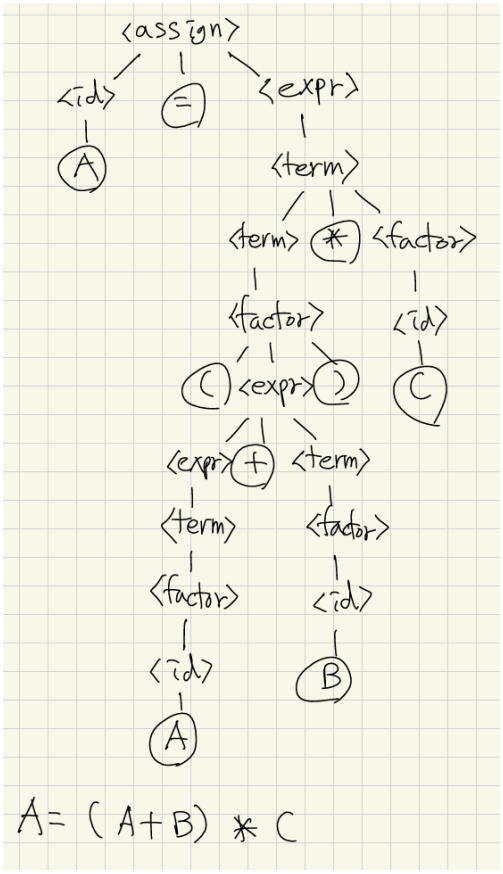
Programming Languages Assignment 1

2021312738 소프트웨어학과 김서환(Kim Seo Hwan)

Q1. 1) A = (A + B) * C

(Please understand that it is difficult to draw parse tree in Word program(.docx file), so I drew parse tree by hand on the I-pad and added it as a picture.)

Parse tree :



Leftmost derivation :

<assign> -> <id> = <expr>

-> A = <expr>

-> A = <term>

-> A = <term> * <factor>

-> A = <factor> * <factor>

-> A = (<expr>) * <factor>

-> A = (<expr> + <term>) * <factor>

-> A = (<term> + <term>) * <factor>

-> A = (<factor> + <term>) * <factor>

-> A = (<id> + <term>) * <factor>

-> A = (A + <term>) * <factor>

-> A = (A + <factor>) * <factor>

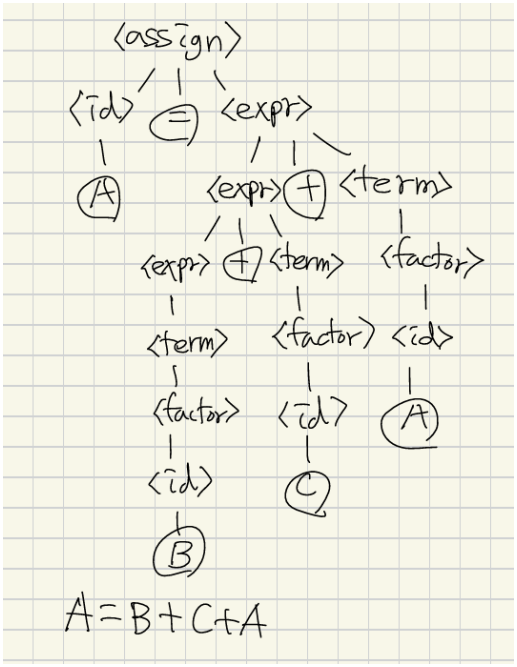
-> A = (A + <id>) * <factor>

-> A = (A + B) * <factor>

-> A = (A + B) * <id>

-> A = (A + B) * C

2) A = B + C + A

Parse tree : 

Leftmost derivation :

<assign> -> <id> = <expr>

-> A = <expr>

-> A = <expr> + <term>

-> A = <expr> + <term> + <term>

-> A = <term> + <term> + <term>

-> A = <factor> + <term> + <term>

-> A = <id> + <term> + <term>

-> A = B + <term> + <term>

-> A = B + <factor> + <term>

-> A = B + <id> + <term>

-> A = B + C + <term>

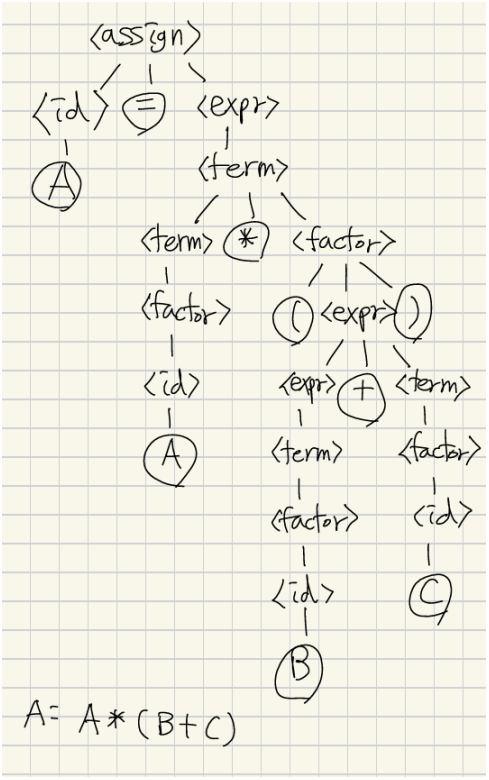
-> A = B + C + <factor>

-> A = B + C + <id>

-> A = B + C + A

3) A = A * (B + C)

Parse tree :



Leftmost derivation :

<assign> -> <id> = <expr>

-> A = <expr>

-> A = <term>

-> A = <term> * <factor>

-> A = <factor> * <factor>

-> A = <id> * <factor>

-> A = A * <factor>

-> A = A * (<expr>)

-> A = A * (<expr> + <term>)

-> A = A * (<term> + <term>)

-> A = A * (<factor> + <term>)

-> A = A * (<id> + <term>)

-> A = A * (B + <term>)

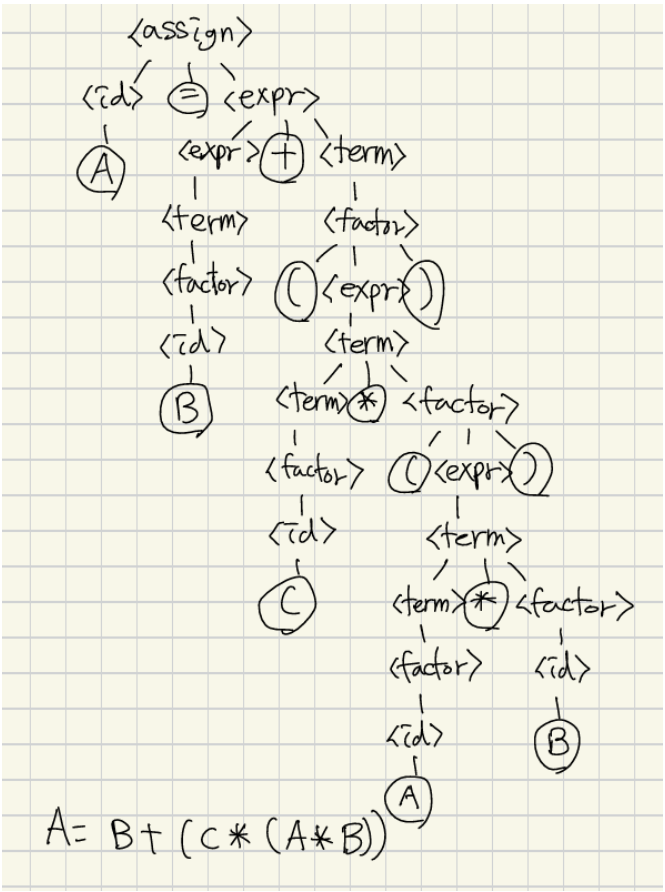
-> A = A * (B + <factor>)

-> A = A * (B + <id>)

-> A = A * (B + C)

4) A = B + (C * (A * B))

Parse tree :



Leftmost derivation :

<assign> -> <id> = <expr>

-> A = <expr>

-> A = <expr> + <term>

-> A = <term> + <term>

-> A = <factor> + <term>

-> A = <id> + <term>

-> A = B + <term>

-> A = B + <factor>

-> A = B + (<expr>)

-> A = B + (<term>)

-> A = B + (<term> * <factor>)

-> A = B + (<factor> * <factor>)

-> A = B + (<id> * <factor>)

-> A = B + (C * <factor>)

-> A = B + (C * (<expr>))

-> A = B + (C * (<term>))

-> A = B + (C * (<term> * <factor>))

-> A = B + (C * (<factor> * <factor>))

-> A = B + (C * (<id> * <factor>))

-> A = B + (C * (A * <factor>))

-> A = B + (C * (A * <id>))

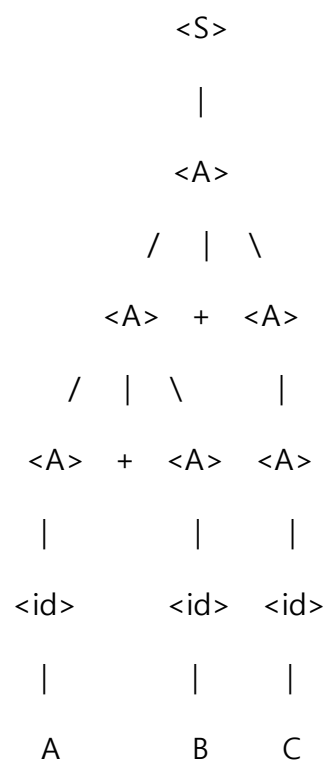
-> A = B + (C * (A * B))

Q2.

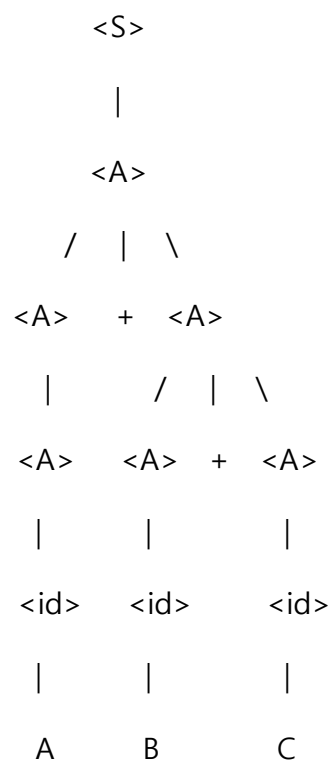
Assume that "A + B + C"

To prove that BNF grammar in Q2 is ambiguous, I should show that there are two or more parse trees for this grammar.

Parse tree 1)



Parse tree 2)



Thus, both Parse trees result in the same final expression "A + B + C". But, the difference in parse tree structures means that the grammar does not enforce a unique way of parsing. In other words, BNF grammar in Q2 is ambiguous.

Q3.

The grammar modified by adding unary minus (-) operator and power (^) operator that have right associativity is as follows.

<Original grammar>

<assign> -> <id> = <expr>

<id> -> A | B | C

<expr> -> <expr> + <term> | <term>

<term> -> <term> * <factor> | <factor>

<factor> -> (<expr>) | <id>

<Modified grammar>

<assign> -> <id> = <expr>

<id> -> A | B | C

<expr> -> <expr> + <term> | <term>

<term> -> <term> * <power> | <power>

<power> -> <unary> ^ <power> | <unary>

<unary> -> -<unary> | <factor>

<factor> -> (<expr>) | <id>