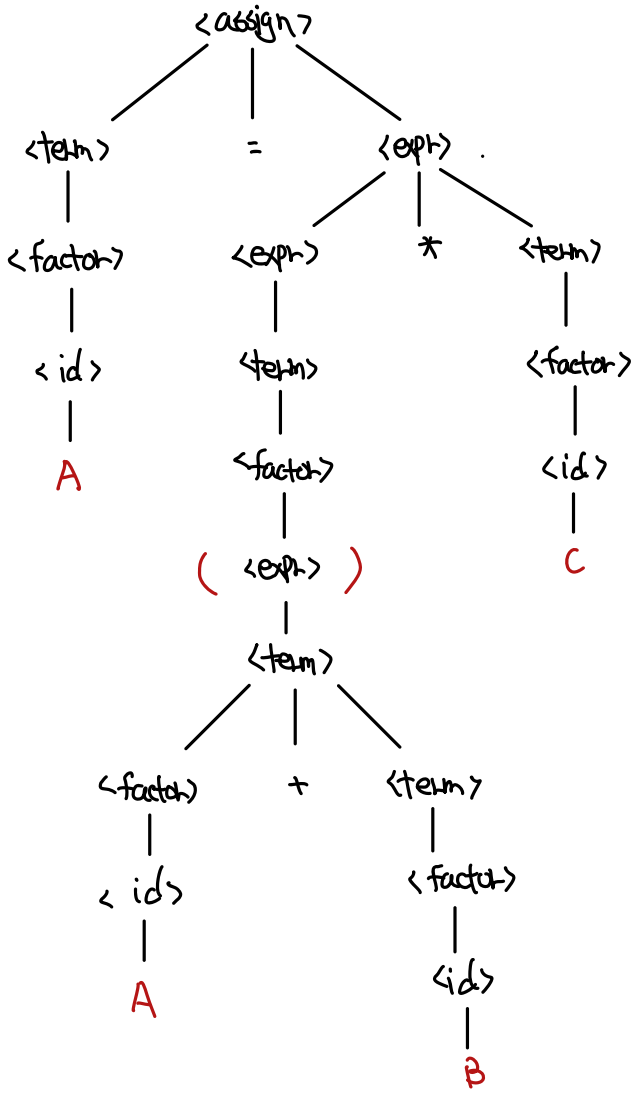
1. A = (A + B) * C



---

2. <assign> → <id> = <expr>

<expr> → <expr> * <term> | <term>

<term> → <factor> + <term> | <factor>

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

<id> → A | B | C

3.

Input : 0 OR 1 AND 1 == True

<compare> → <bool expr> == <bool expr>

<bool expr> AND <bool expr> == <bool expr>

<bool expr> OR <bool expr> AND <bool expr> == <bool expr>

<bool> OR <bool expr> AND <bool expr> == <bool expr>

<bool value> OR <bool expr> AND <bool expr> == <bool expr>

0 OR <bool expr> AND <bool expr> == <bool expr>

0 OR <bool> AND <bool expr> == <bool expr>

0 OR <bool value> AND <bool expr> == <bool expr>

0 OR 1 AND <bool expr> == <bool expr>

0 OR 1 AND <bool> == <bool expr>

0 OR 1 AND <bool value> == <bool expr>

0 OR 1 AND 1 == <bool>

0 OR 1 AND 1 == <bool value>

0 OR 1 AND 1 == True

<compare> → <bool expr> == <bool expr>.

<bool expr> OR <bool expr> == <bool expr>

<bool> OR <bool expr> == <bool expr>

<bool value> OR <bool expr> == <bool expr>

0 OR <bool expr> == <bool expr>

0 OR <bool expr> AND <bool expr> == <bool expr>

0 OR <bool> AND <bool expr> == <bool expr>

0 OR <bool value> AND <bool expr> == <bool expr>

0 OR 1 AND <bool expr> == <bool expr>

0 OR 1 AND <bool> == <bool expr>

0 OR 1 AND <bool value> == <bool expr>

0 OR 1 AND 1 == <bool>

0 OR 1 AND 1 == <bool value>

0 OR 1 AND 1 == True

---

4. a) A Java class definition header statement :

Java example : Public class main {int x=5}
<class declaration> → <class modifiers> class <id> {class body}

<class declaration> → <class modifiers> class <identifier> <class body>

<class modifiers> → <class modifier> | <class modifiers> <class modifier>

<class modifier> → Public | abstract | final

<identifier> → <Letter> | <digit>

<Letter> → a|... | z | A .... | Z

<digit> → 0|... | 9

<class body> → <Type> <assign>

<Type> → int | string | char ....

<assign> → <identifier> = <expr>

<expr> → <identifier>

b) A C switch statement

C sample : switch (grade) {case (grade >90) : Printf ('A'); break ;}

<selection-statement> → switch (<expr>) {<labeled-statement>}

<selection-statement> → switch (<expr>){<labeled-statement>}

<expr> → <type> <id>

<type> → <char>|<string>|<int> ...

<id> → <Letter>|<digit>.

<Letter> → a|...|z|A|...|Z.

<digit> → 0...|..9

<labeled-statement> → case <constant-expr>: <statement>.

<constant-expr> → <relational-expr>

<relational-expr> → <relational-expr> >= <digit>

<statement> → <expr> | <jump-statement>

<jump-statement> → break;

c) A C union definition

union definition : union union_name { datatype field_name; datatype field_name;};

<struct-or-union-specifier> → <struct-or-union> <Letter> {<expr>}

<struct-or-union-specifier> → <struct-or-union> <id> {<expr>}

<struct-or-union> → struct | union.

<id> → <Letter>|<digit>.

<Letter> → a|...|z|A|...|Z.

<digit> → 0...|..9

<expr> → <type> <Letter>