Chapter 3 - Homework and Classwork

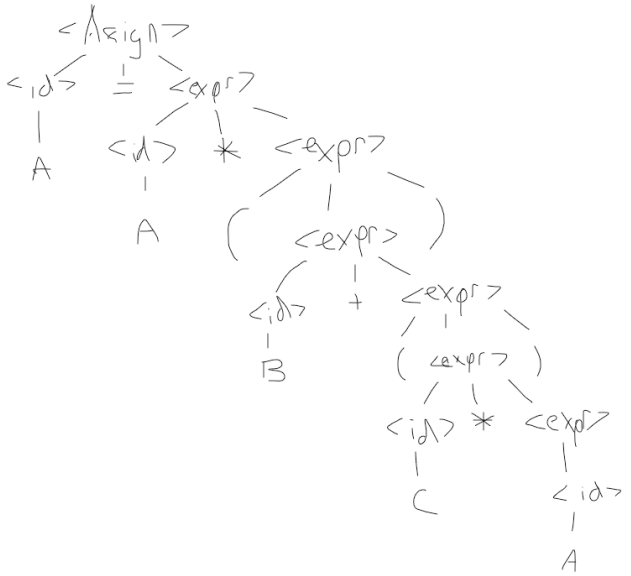
3. Rewrite the BNF of Example 3.4 to give + precedence over * and force + to be right associative.

<assign> => <id> = <expr>
<id> => A | B | C
<expr> => <expr> * <term>
| <term>
<term> => <factor> + <term>
| <factor>
<factor> => (<expr>)
| <id>

6. Using the grammar in Example 3.2, show a parse tree and a leftmost derivation for each of the following statements:

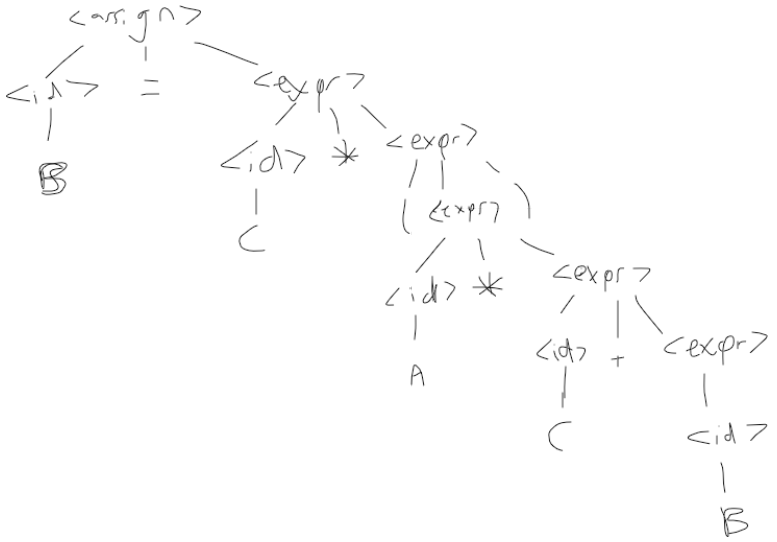
A. A = A * (B + (C * A))

<assign> => <id> = <expr>
=> A = <expr>
=> A = <id> * <expr>
=> A = A * <expr>
=> A = A * (<expr>)
=> A = A * (<id> + <expr>)
=> A = A * (B + <expr>)
=> A = A * (B + (<id> * <expr>))
=> A = A * (B + (C * <expr>))
=> A = A * (B + (C * <id>))
=> A = A * (B + (C * A))



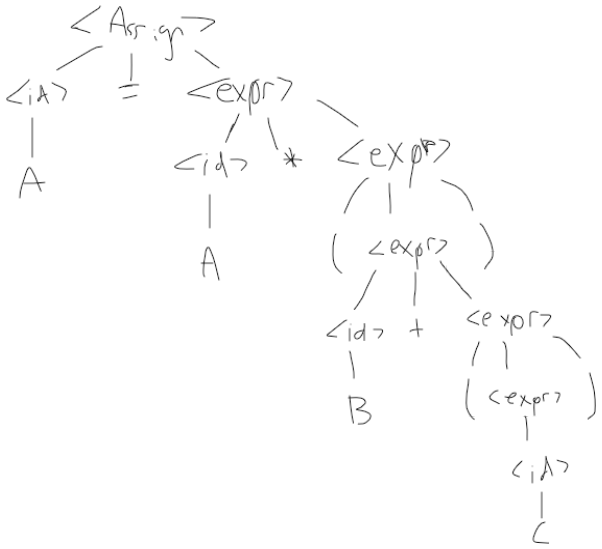
B. B = C * (A * C + B)

<assign> => <id> = <expr>
=> B = <expr>
=> B = <id> * <expr>
=> B = C * <expr>
=> B = C * (<expr>)
=> B = C * (<id> * <expr>)
=> B = C * (A * <expr>)
=> B = C * (A * <id> + <expr>)
=> B = C * (A * C + <expr>)
=> B = C * (A * C + <id>)
=> B = C * (A * C + B)



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

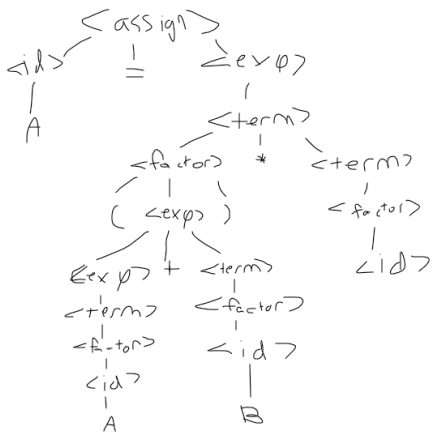
<assign> => <id> = <expr>
=> A = <expr>
=> A = <id> * <expr>
=> A = A * <expr>
=> A = A * (<expr>)
=> A = A * (<id> + <expr>)
=> A = A * (B + <expr>)
=> A = A * (B + (<expr>))
=> A = A * (B + (<id>))
=> A = A * (B + (C))



7. Using the grammar in Example 3.4, show a parse tree and a leftmost derivation for each of the following statements:

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

<assign> => <id> = <expr>
=> A = <expr>
=> A = <term>
=> A = <factor> * <term>
=> A = (<expr>) * <term>
=> A = (<expr> + <term>) * <term>
=> A = (<term> + <term>) * <term>
=> A = (<factor> + <term>) * <term>
=> A = (<id> + <term>) * <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



B. **A = B + C + A**

<assign> => <id> = <exp>

=> A = <exp> + <term>

=> A = <exp> + <term> + <term>

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

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

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

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

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

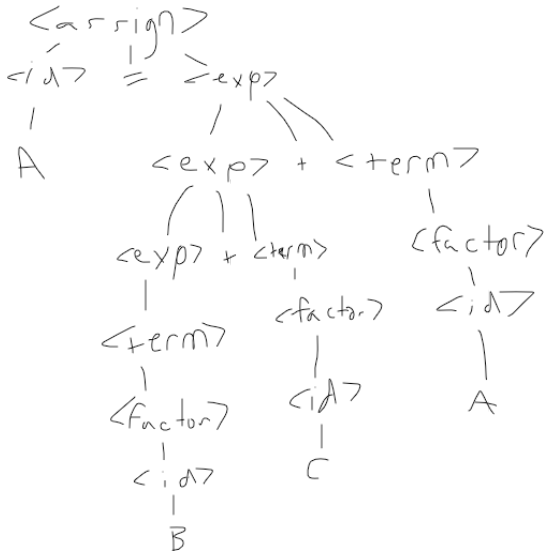
=> A = B + <id> + <term>

=> A = B + C + <term>

=> A = B + C + <factor>

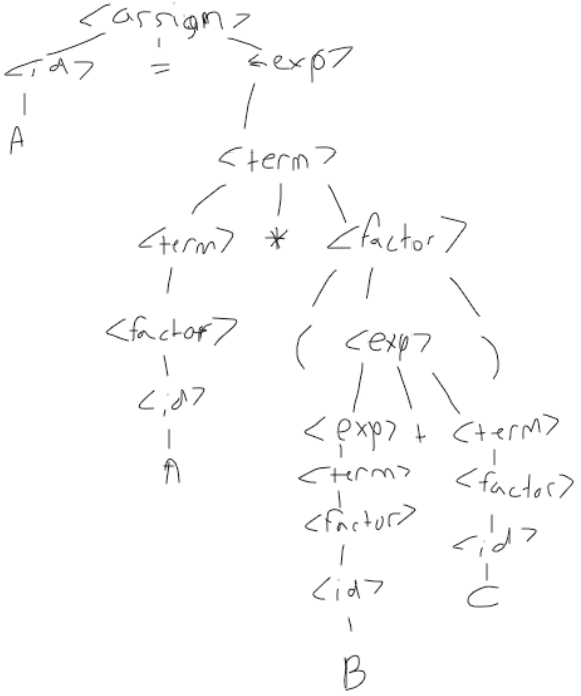
=> A = B + C + <id>

=> A = B + C + A



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

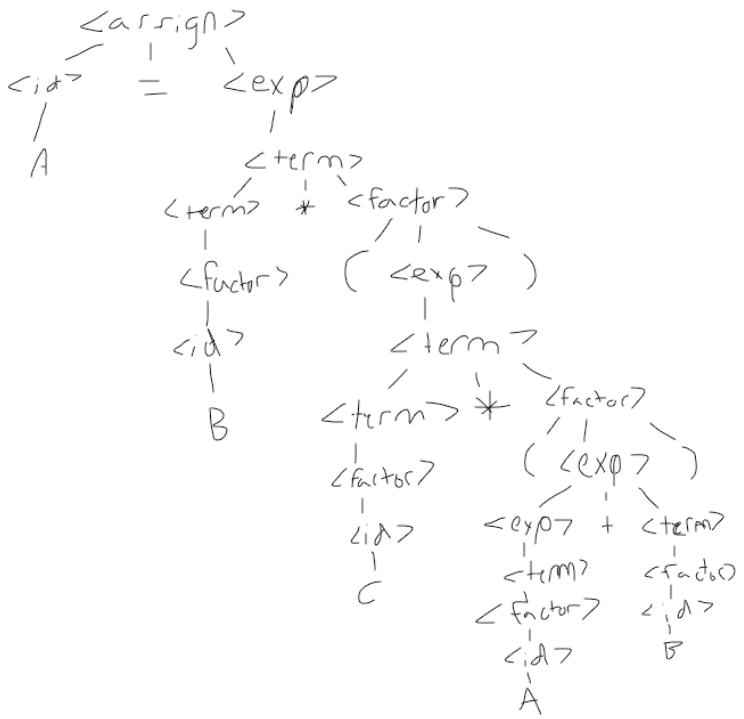
<assign> => <id> = <exp>
=> A = <exp>
=> A = <term>
=> A = <term> * <factor>
=> A = <factor> * <factor>
=> A = <id> * <factor>
=> A = A * <factor>
=> A = A * (<exp>)
=> A = A * (<exp> + <term>)
=> A = A * (<term> + <term>)
=> A = A * (<factor> + <term>)
=> A = A * (<id> + <term>)
=> A = A * (B + <term>)
=> A = A * (B + <factor>)
=> A = A * (B + <id>)
=> A = A * (B + C)



D. **A = B * (C * (A + B))**

<assign> => <id> = <exp>
=> A = <exp>
=> A = <term>
=> A = <term> * <factor>
=> A = <factor> * <factor>
=> A = <id> * <factor>

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



11. Consider the following grammar:

- **<S> → <A> a <B> b**
- **<A> → <A> b | b**
- **<B> → a <B> | a**

Which of the following sentences are in the language generated by this grammar?

A. baab in the language

<a>a<b>b
ba<b>b
ba<b>b
baab

B. bbbab not in the language. End up with bbba<B>b.

C. bbaaaaa not in the language. Has to end in b.

D. bbaab in the language

<a>a<b>b
<a>ba<b>b
bba<b>b
bbaab

12. Consider the following grammar:

- **<S> → a <S> c <B> | <A> | b**
- **<A> → c <A> | c**
- **<B> → d | <A>**

Which of the following sentences are in the language generated by this grammar?

A. abcd

a<s>c<b>|<a>|b
abc<b>
abcd Works

B. acccbd

a<s>c<b>|<a>|b
a<a>c<b>
a<a>cc<b>
accc<b> Does not work. Can't get "b" from <b>

C. acccbcc

a<s>c<b>|<a>|b
a<a>c<b>
a<a>cc<b>
accc<b> Does not work. Won't be able to add in a "b" before cc at the end.

D. acd

a<s>c<b>|<a>|b
a<s>c<b> Does not work. Can't have "a" in front without having 4 letters

E. accc

a<s>c<b>|<a>|b
a<a>c<b>

acc<b>

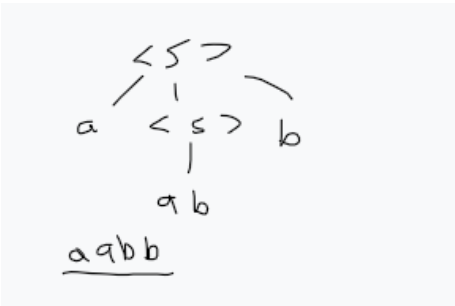
acc<a>

accc Works

13. Write a grammar for the language consisting of strings that have *n* copies of the letter a followed by the same number of copies of the letter b, where *n* > 0. For example, the strings ab, aaaabbbb, and aaaaaaaabbbbbbbb are in the language but a, abb, ba, and aaabb are not.

S => a <S> b | a b

14. Draw parse trees for the sentences aabb and aaaabbbb, as derived from the grammar of Problem 13.



16. Convert the BNF of Example 3.3 to EBNF.

Original from 3.3:

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

<id> => A | B | C

<expr> => <expr> + <expr>

| <expr> * <expr>

| (<expr>)

| <id>

Converted to EBNF:

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

<id> => A | B | C

<expr> => <expr> (+ | *) <expr>

| (<expr>)

| <id>

Class Activity - 1/25/23

23. Compute the weakest precondition for each of the following assignment statements and postconditions:

A. a = 2 * (b - 1) - 1 {a > 0}

{a>0} = 2 * (b-1) - 1
0 < 2 * (b-1) - 1
2b - 3 > 0
2b > 3
b > 3/2

B. b = (c + 10) / 3 {b > 6}
(c+10) / 3 > 6
c+10 > 18
c > 8

C. a = a + 2 * b - 1 {a > 1}
a + 2 * b - 1 > 1
2 * b > 2 - a
b > 1 - a / 2

D. x = 2 * y + x - 1 {x > 11}
2 * y + x - 1 > 11
2 * y + x > 12