YASHASHWINI GUPTA
yg1568@nyu.edu

# PROGRAMMING LANGUAGES

## PROBLEM SET - ASSIGNMENT 1 (CHAPTER 3)

Q3. Rewrite the BNF of example 3.4 to give + precedence over * and force + to be right associative.

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

Q4. Rewrite the BNF of example 3.4 to add ++ and -- unary operators of Java.

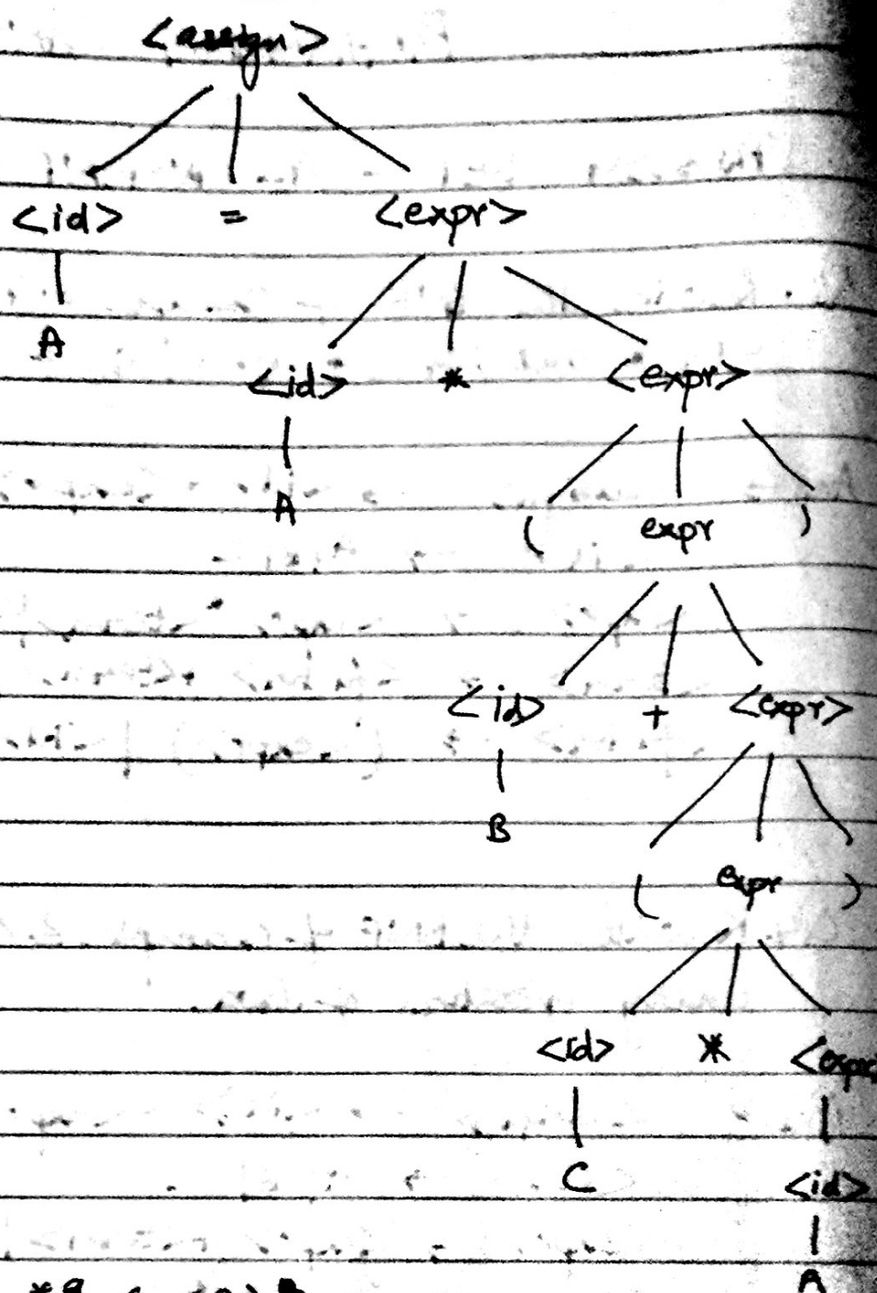
Ans→
<assign> → <id> = <expr>
<id> → A | B | C
<expr> → <expr> + <term> | <term>
<term> → <term> * <factor> | <factor>
<factor> → (<expr>) | <id> | <id>++ | <id>--

Q6. Using the grammer in example 3.2 show a parse tree and a leftmost derivation for each of the following statements.

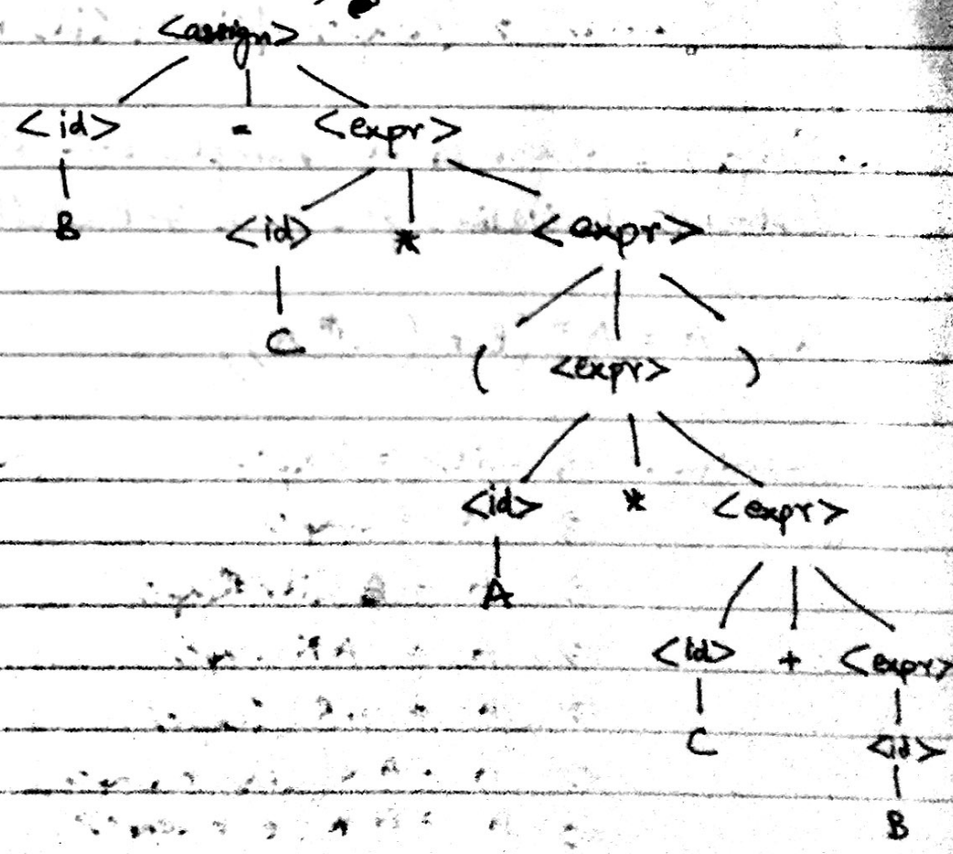
a) A = A * (B + (C * A))

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

<assign> ⇒ 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 ) )



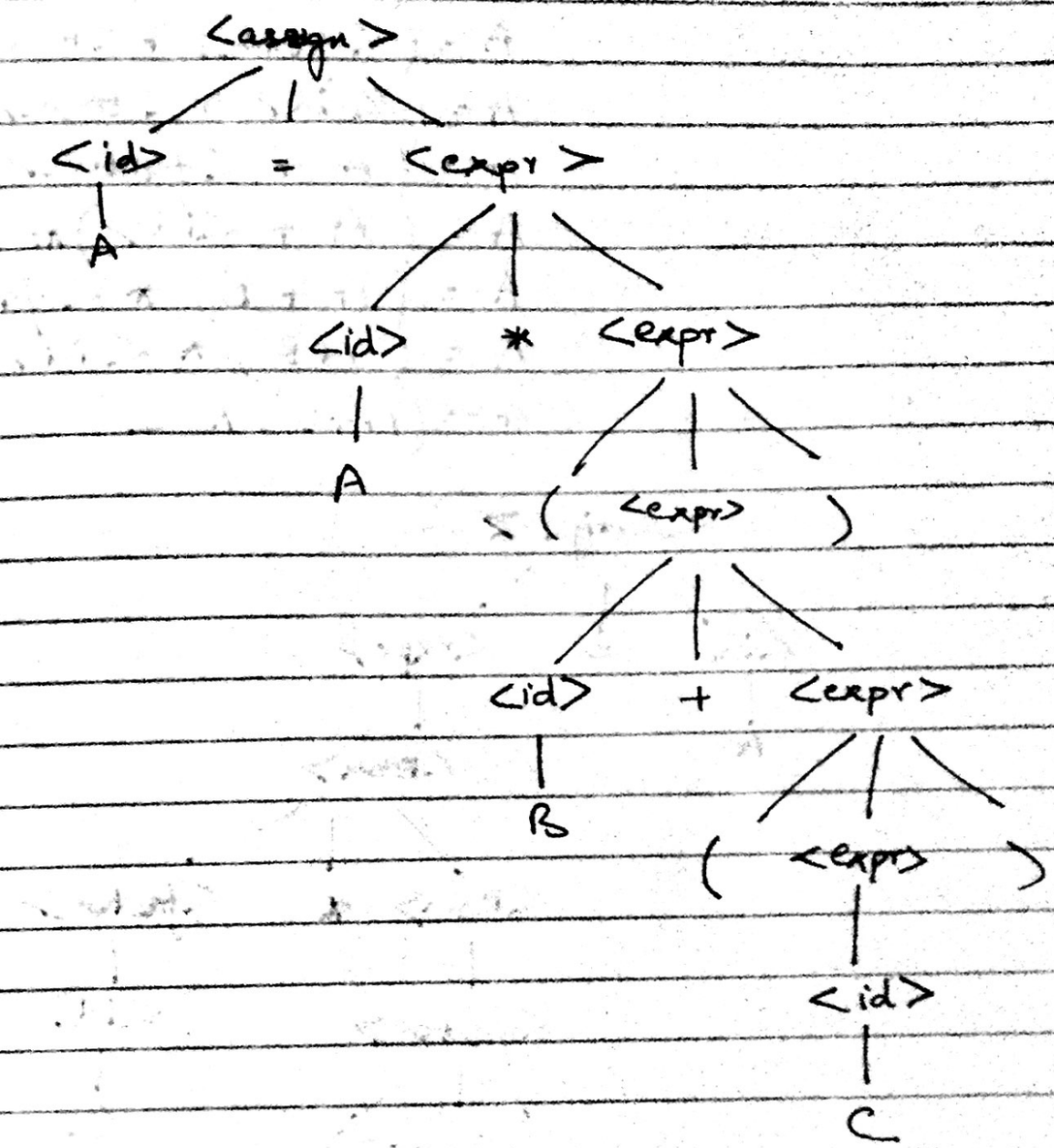
(b) B = C * (A * C + B)

<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))

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




Q7. Using the grammer in example 3.4, show a parse tree and a left most derivation for each.

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

<assign> ⇒ <id> = <expr>

⇒ A = <expr>

⇒ A = [illegible] <term>

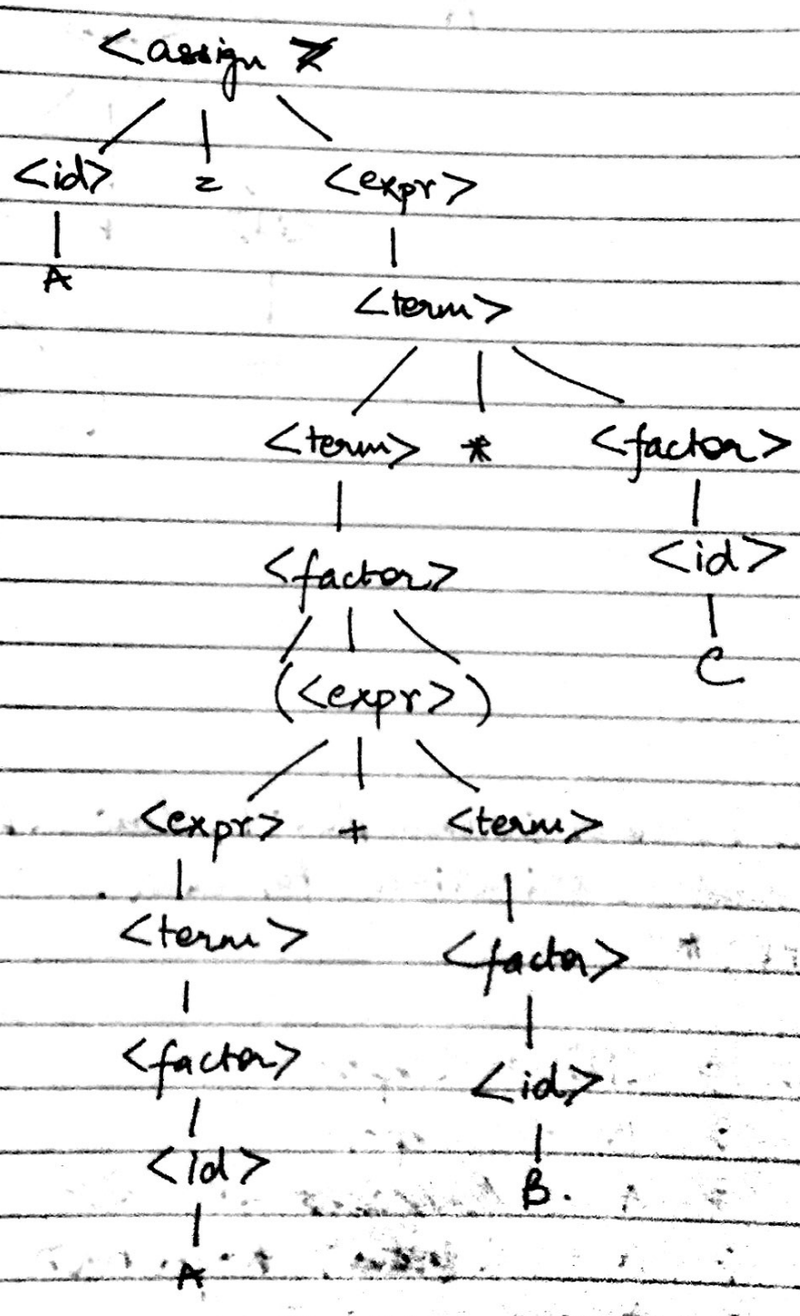
⇒ A = <[illegible]> * <factor>

~~⇒ A = (<expr>)~~

⇒ A = <factor> * <factor>

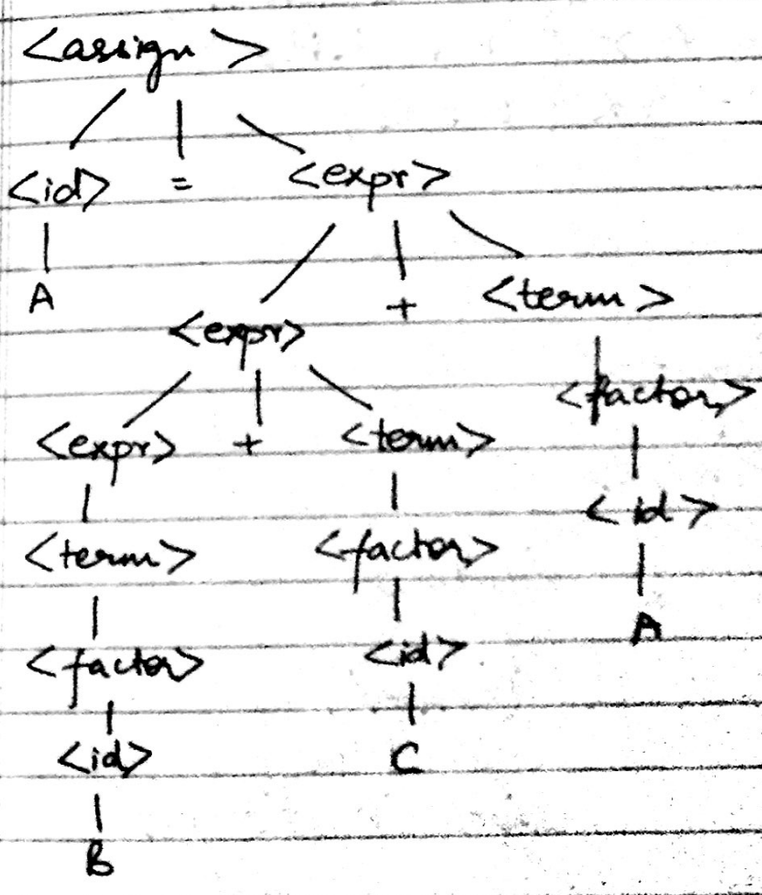
⇒ A = (<expr>) * <factor>

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



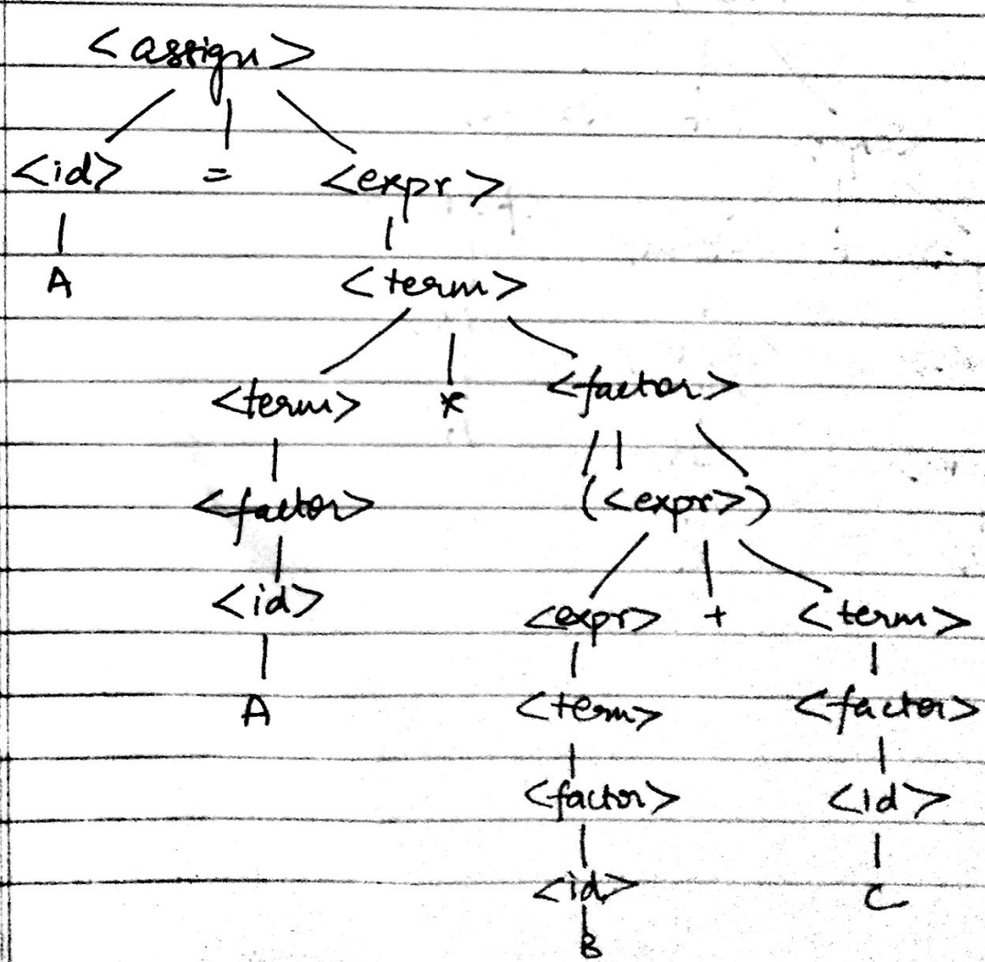
Q7.(b) A = B + C + A

<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



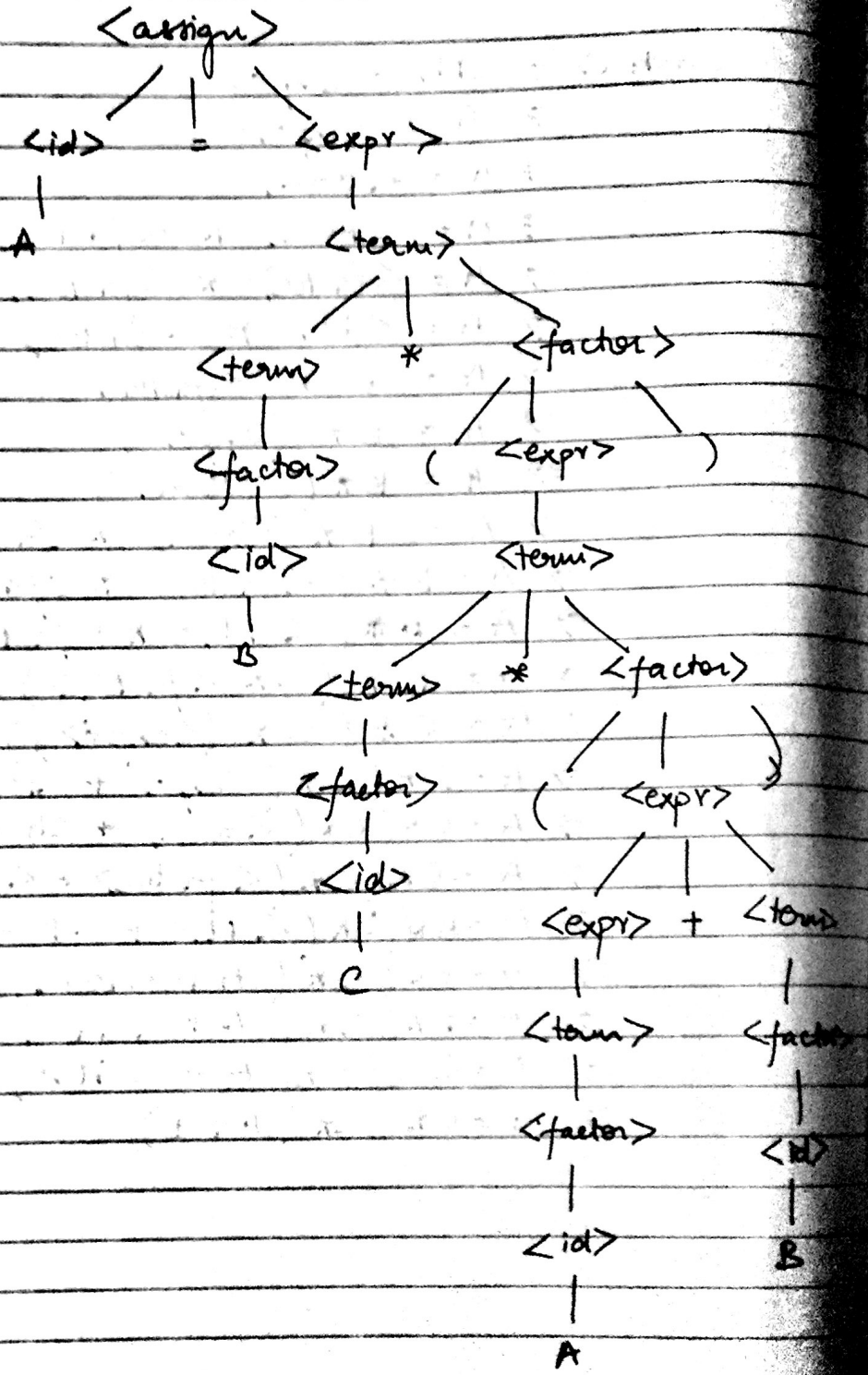
(c) $A = A * (B + C)$

<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)



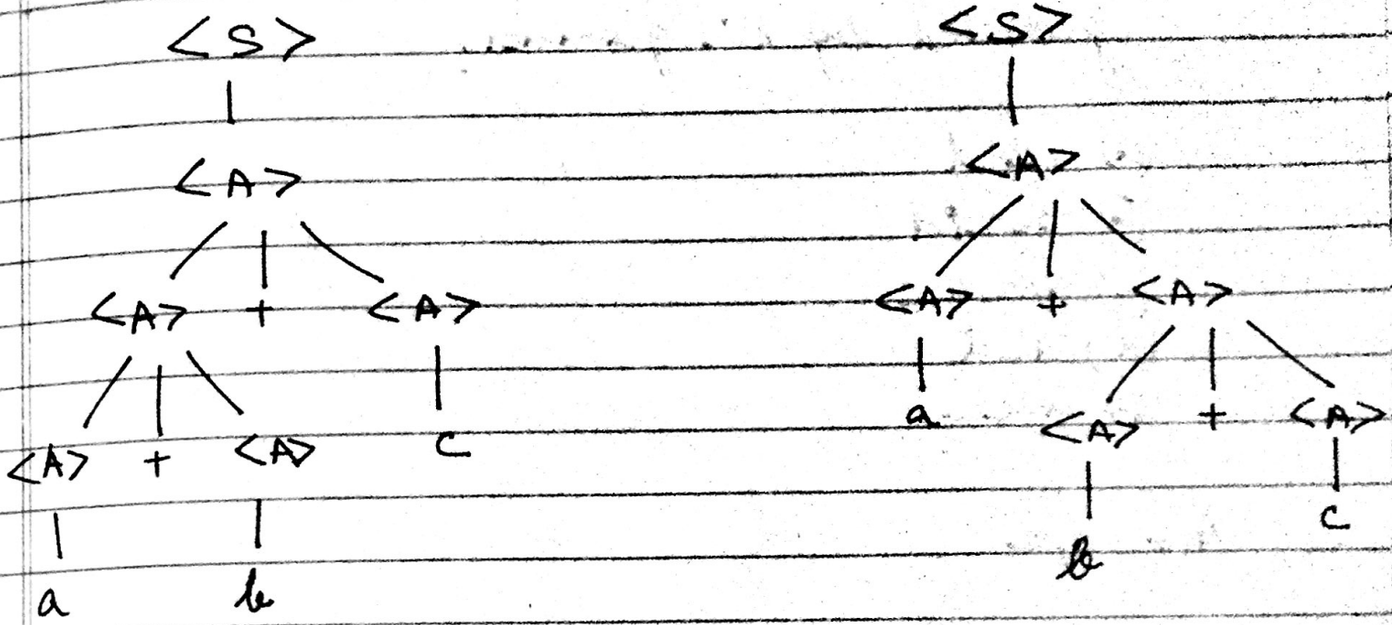
(d) A = B * ( C * (A+B))

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


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

Q8. Prove that the following grammer is ambiguous.

<S> → <A>
<A> → <A> + <A> | <id>
<id> → a | b | c




Here, for a+b+c there are two possible parse trees generated from the grammer.

Hence the grammer is ambigous.

Q11. Consider the following grammer:

<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 grammer?

(a) baab
(b) bbbab
(c) bbaaaa
(d) bbaab

Ans. a, d

Q12. Consider the following grammer:

<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 grammer?

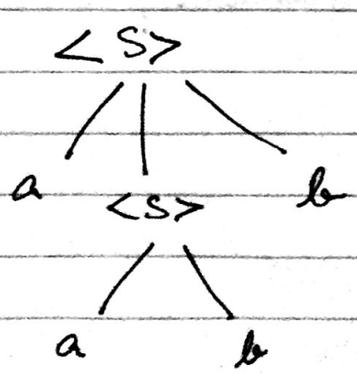
(a) abcd
(b) acccbd
(c) acccbcc
(d) acd
(e) accc

Ans. a, c

Q13. 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, aaabbbbb and aaaaaaaabbbbbbbbb are in the language but a, abbb, ba, and aaabb are not.

<S> → ab | a<S>b

Q14. Draw parse trees for the sentences aabb, and aaabbbbb, as derived from the grammer of Problem 13




Parse tree for aabb

Parse tree for aaaabbbb