**Assignment 2**

Problem set questions:

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

This is the original tree from example 3.4

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

<id> -> A | B | C

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

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

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

To be rewrite it to give + precedence over \* and force + to be right associative, the tree would look like this

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

<id> -> A | B | C

<expr> -> <expr> \* <term> | <term>

<term> -> <term> + <factor> | <factor>

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

**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

b. A = B + C + A

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

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

example 3.4 is:

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

<id> -> A | B | C

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

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

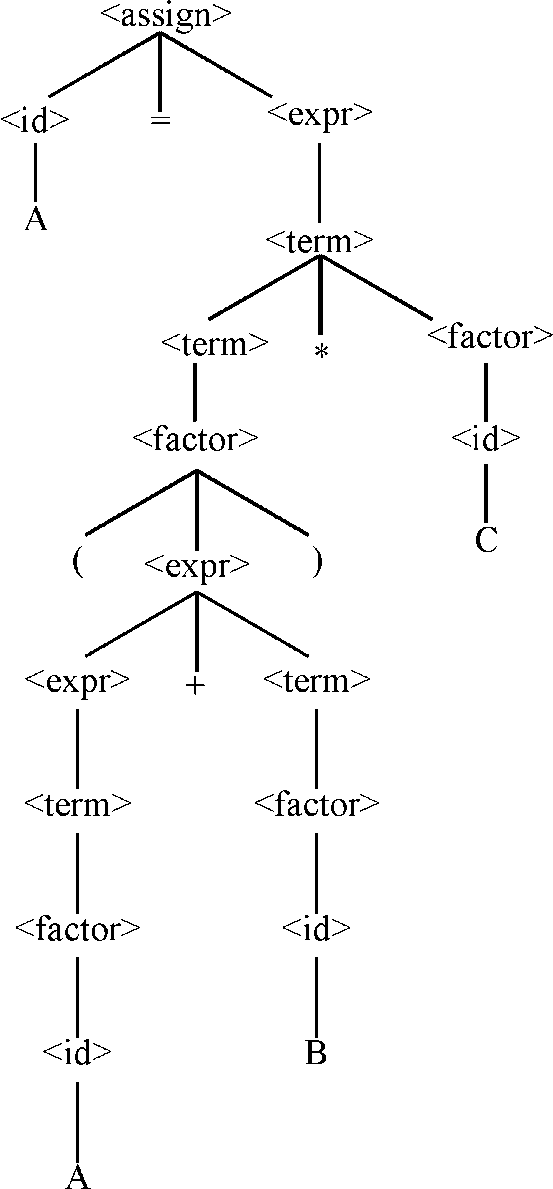
<factor> -> ( <expr> )| <id>

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

Left most Derivation for the statement:

<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

**Parse tree:**

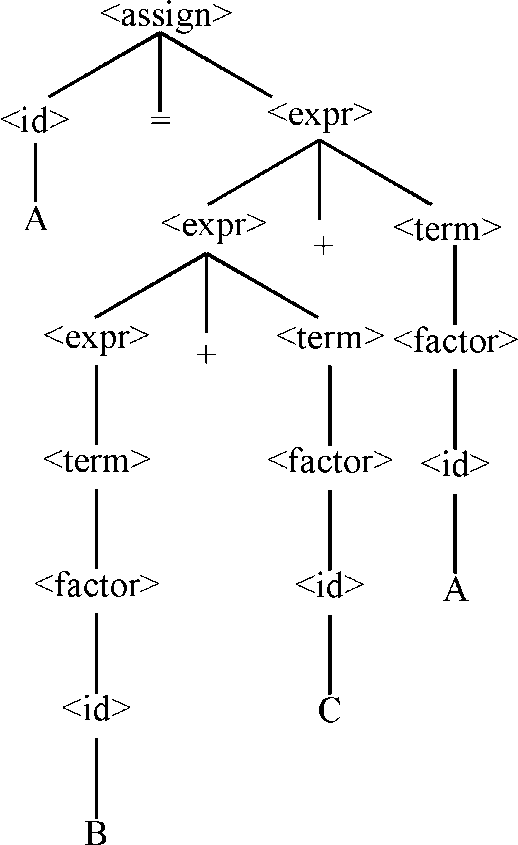
1. A = B + C + A

b. Left most Derivation for the statement:

<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

**Parse tree:**

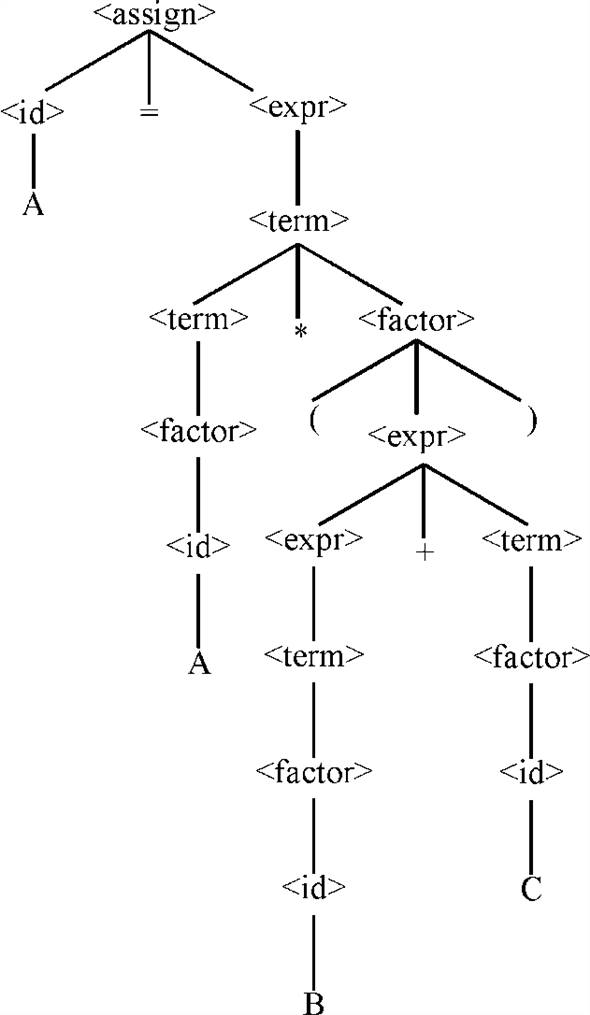
****

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

c. Left most Derivation for the statement:

<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>)

**Parse tree:**

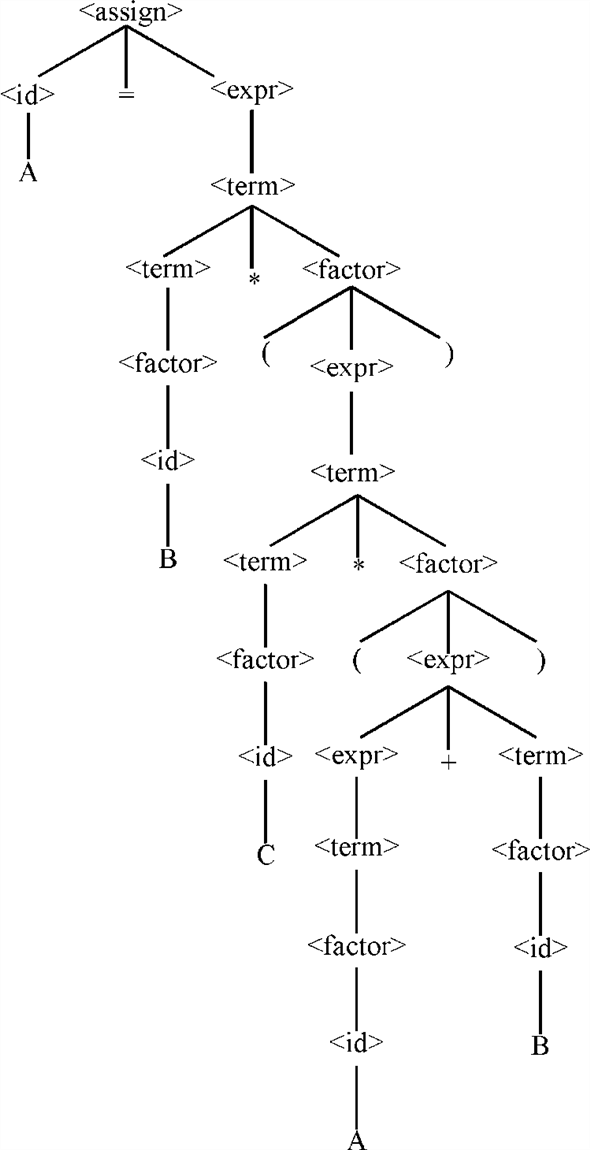
1. A = B \* (C \*(A + B))

d. Left most Derivation for the statement:

<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)

**Parse tree:**

****

**8. Prove that the following grammar is ambiguous:**

<S> → <A>

<A> →<A> + <A> | <id>

<id> → a | b | c

The grammar would be ambiguous if there is more than one tree to represent the same expression

If we compute the leftmost derivations from that expression above, we get two answers:

First:

* <A> + <A>
* <A> + <A> + <A>
* <id> + <A> + <A>
* a + <A> + <A>
* a + <id> + <A>
* a + b + <A>
* a + b + <id>
* a + b + c

Second:

* <A> + <A>
* <id> + <A>
* a + <A>
* a + <A> + <A>
* a + <id> + <A>
* a + b + <A>
* a + b + <id>
* a + b + c

This proves that the expression above is ambiguous

**9. Modify the grammar of Example 3.4 to add a unary minus operator that**

**has higher precedence than either + or \*.**

The modified solution is, and change is made in third line.

<assign> → <id> = <expr>

<id> → A|B|C

<expr> → <expr> + <term> | <term> | - <factor>

<term> → <term> \* <factor> | <factor>

<factor> → (<expr>) | <id>

**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

b. bbbab

c. bbaaaaa

d. bbaab

Our language must have the following rules

At least one or more ‘b’

One ‘a’

At least one or more ‘a’

One ‘b’

The only sentences that following this rules are “baab” and “bbaaaaa”

**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.**

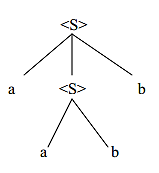
The grammar would be:

<S> -> ab | a <S> b

**14. Draw parse trees for the sentences aabb and aaaabbbb, as derived from**

**the grammar of Problem 13.**

**Parse tree for the sentence** **aabb:**



**Parse tree for the sentence** **aaaabbbb:**

