

Problem 3.3

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

EXAMPLE 3.4**An Unambiguous Grammar for Expressions**

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

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

Problem 3.4

4. Rewrite the BNF of Example 3.4 to add the `++` and `--` unary operators of Java.

EXAMPLE 3.4**An Unambiguous Grammar for Expressions**

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

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

Problem 3.7

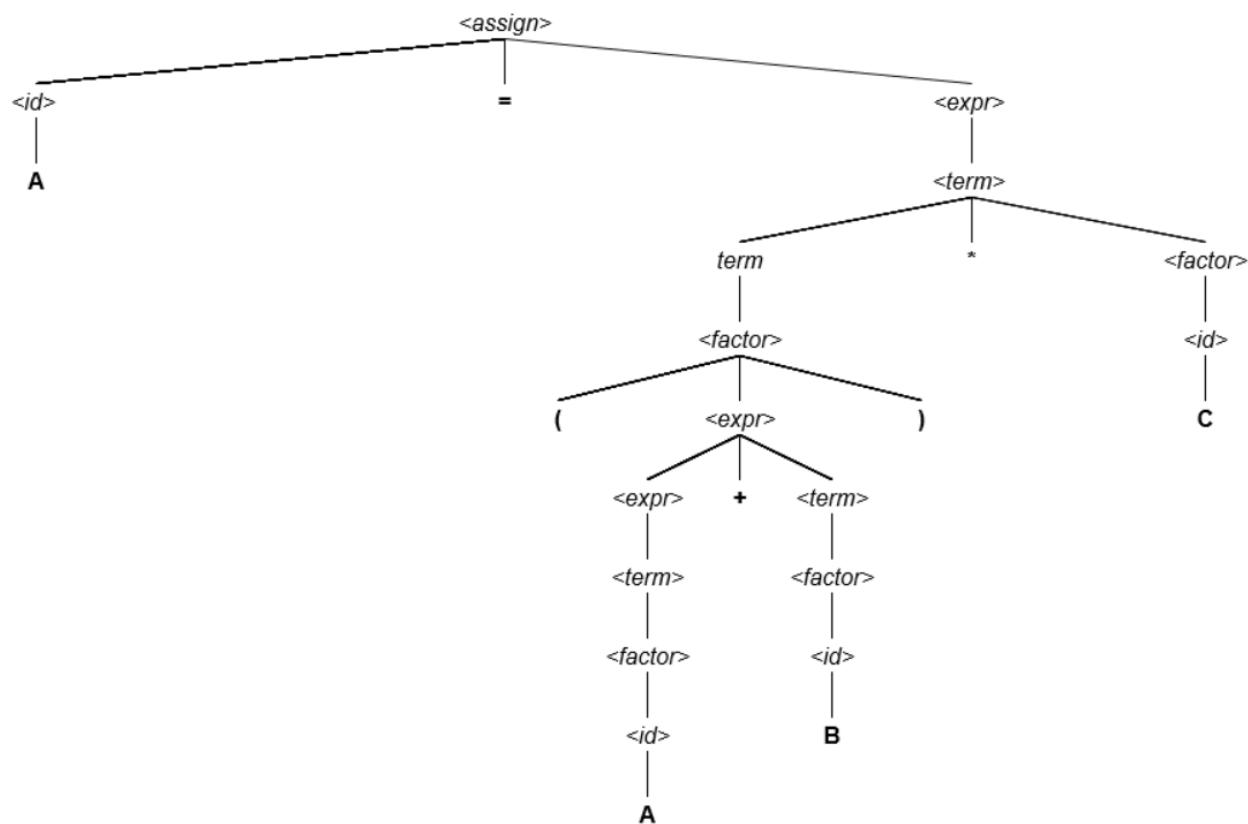
7. Using the grammar in Example 3.4, show a parse tree and a leftmost derivation for each of the following statements:
- $A = (A + B) * C$
 - $A = B + C + A$
 - $A = A * (B + C)$
 - $A = B * (C * (A + B))$

EXAMPLE 3.4**An Unambiguous Grammar for Expressions**

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

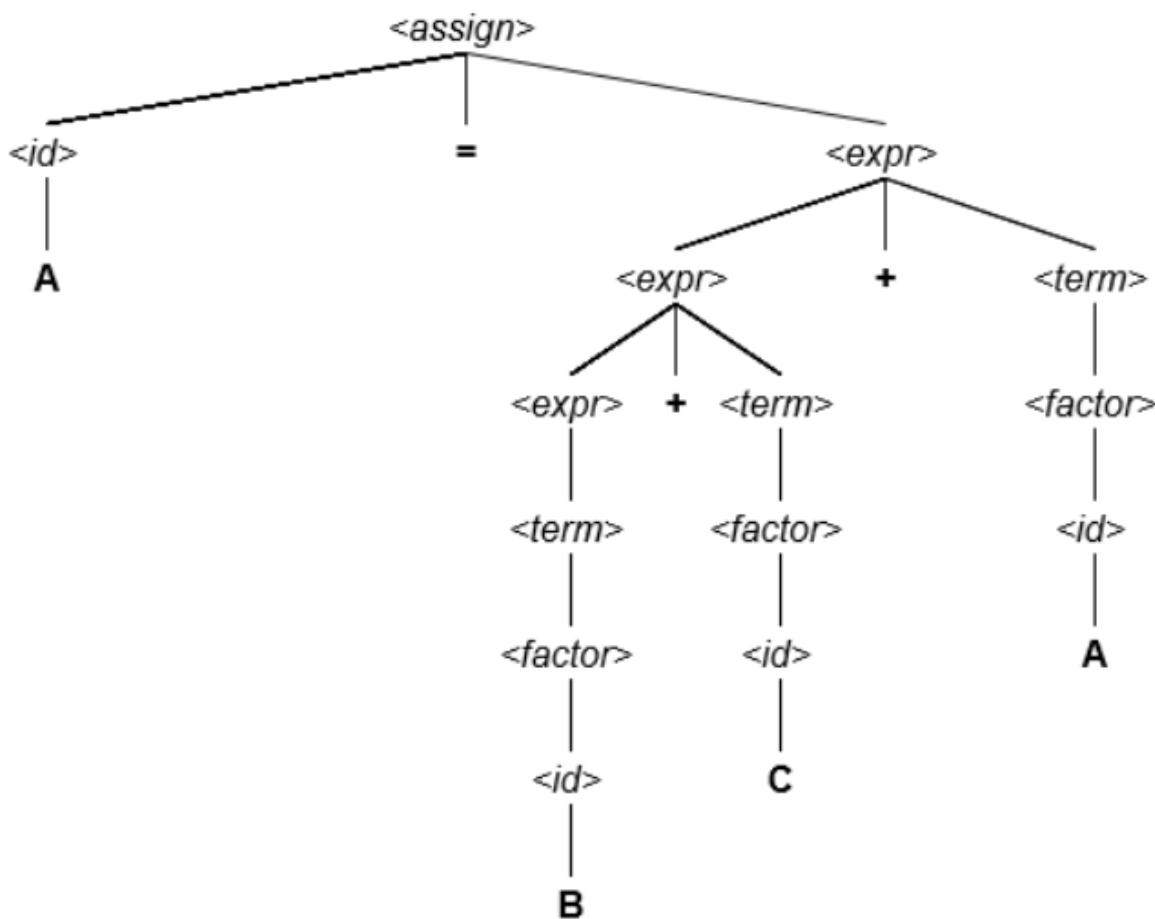
7(a) $A = (A + B) * C$

```
<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
```



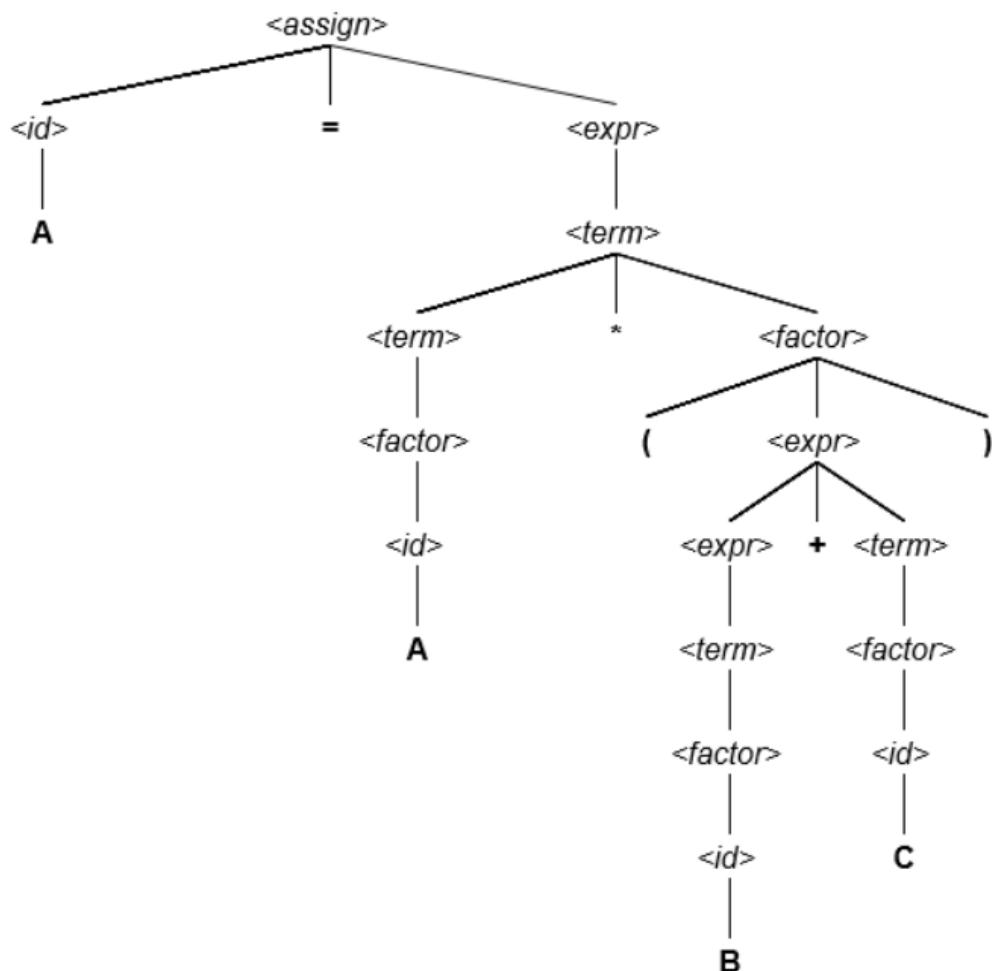
7(b) A = B + C + A

```
<assign> → <id> = <expr>
          → A = <expr>
          → A = <expr> + <term>
          → A = <expr> + <term> + <expr>
          → 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
```



7(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 )
```

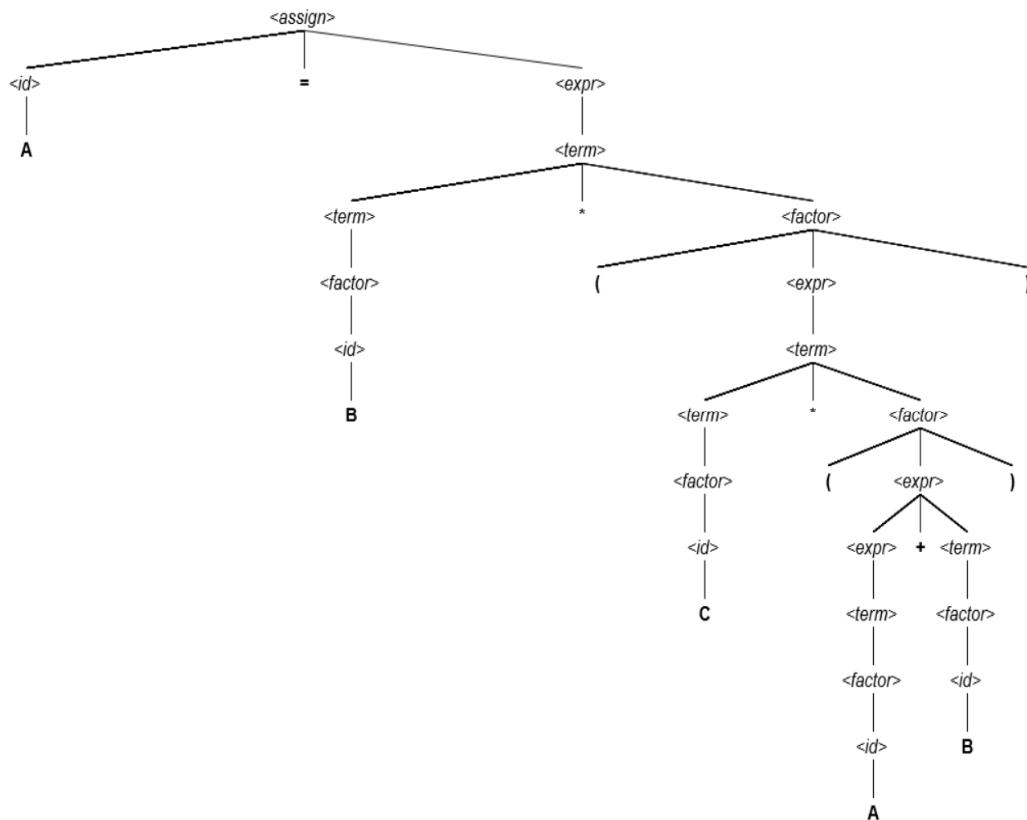


7(d) $A = B * (C * (A + B))$

```

<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 ) )

```



Problem 3.8

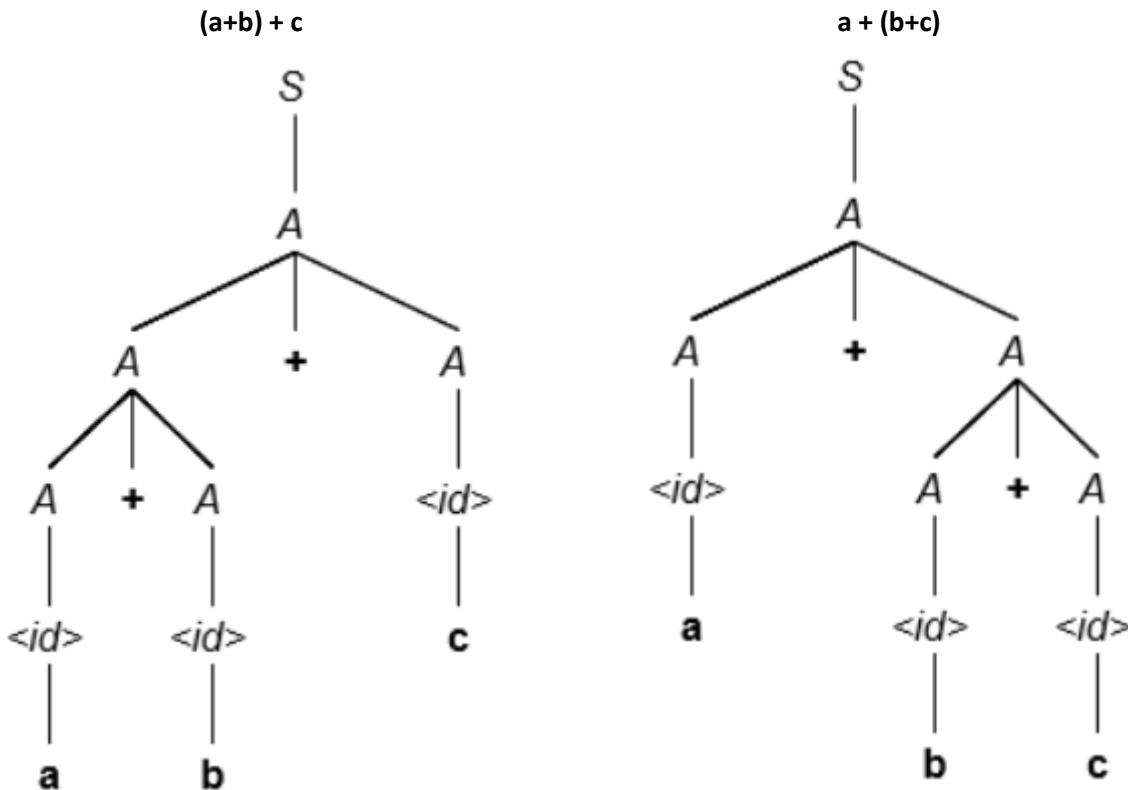
8. Prove that the following grammar is ambiguous:

$$\langle S \rangle \rightarrow \langle A \rangle$$

$$\langle A \rangle \rightarrow \langle A \rangle + \langle A \rangle \mid \langle id \rangle$$

$$\langle id \rangle \rightarrow a \mid b \mid c$$

Example: a + b + c



Problem 3.9

9. Modify the grammar of Example 3.4 to add a unary minus operator that has higher precedence than either + or *.

EXAMPLE 3.4

An Unambiguous Grammar for Expressions

```

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

<assign> → <id> = <expr>

<id> → A | B | C

<expr> → <expr> + <term>
| <term>

<term> → <term> * <factor>
| <factor>

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

Problem 3.10

10. Describe, in English, the language defined by the following grammar:

```

<S> → <A> <B> <C>
<A> → a <A> | a
<B> → b <B> | b
<C> → c <C> | c
  
```

At least one 'a' followed by at least one 'b' followed by at least one 'c'

Problem 3.12

12. Consider the following grammar:

$$\langle S \rangle \rightarrow a \langle S \rangle c \langle B \rangle \mid \langle A \rangle \mid b$$

$$\langle A \rangle \rightarrow c \langle A \rangle \mid c$$

$$\langle B \rangle \rightarrow d \mid \langle A \rangle$$

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

- a. abcd
- b. acccbd
- c. acccbcc
- d. acd
- e. accc

Sentences (a) and (e) only

abcd	acc
<pre> graph TD S1["<S>"] --- a1[a] S1 --- S2["<S>"] S1 --- c1[c] S1 --- B1[""] S2 --- b1[b] B1 --- d1[d] </pre>	<pre> graph TD S1["<S>"] --- a1[a] S1 --- S2["<S>"] S1 --- c1[c] S1 --- B1[""] S2 --- A1["<A>"] A1 --- c2[c] B1 --- A2["<A>"] A2 --- c3[c] </pre>

Since all of the sentences start with 'a', the first rule used must be [a <S> c]. Thus any sentence that begins with 'a' must end with which means that the sentence can end with 'cd' or with one or more 'c's. This rules out (b).

Since <S> generates at least one letter, between the 'a' and the ending combination there must be at least one letter, which rules out (d).

The only way to generate a 'b' is from <S> directly, meaning that the only sentence that can start with 'a' and end with 'bcc' is 'abcc'. This rules out (c).

Problem 3.13

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 aaaaaaaaaabbbbbbbb are in the language but a, abb, ba, and aaabb are not.

$$\langle S \rangle \rightarrow ab \mid a \langle S \rangle b$$
Problem 3.14

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

aabb	aaaabbbb
<pre> graph TD S1["<S>"] --- a1[a] S1 --- S2["<S>"] S1 --- b1[b] S2 --- ab1[ab] </pre>	<pre> graph TD S1["<S>"] --- a1[a] S1 --- S2["<S>"] S1 --- b1[b] S2 --- a2[a] S2 --- S3["<S>"] S2 --- b2[b] S3 --- a3[a] S3 --- S4["<S>"] S3 --- b3[b] S4 --- ab2[ab] </pre>

Problem 3.17

17. Convert the following EBNF to BNF:

$$S \rightarrow A\{bA\}$$

$$A \rightarrow a[b]A$$

$$S \rightarrow A$$

$$| \quad A B$$

$$A \rightarrow a A$$

$$| \quad a b A$$

$$B \rightarrow b A$$

$$| \quad B b A$$