Problem Sets from Chapter three

In class I will do Questions: #2B, #6A, #8, #10, (The I will start on LISP)

For homework you will do questions: #2C, #3, #6B, #7A, #13, #14

This homework will be due before class on Monday September 14th, You can turn it in by hand in lecture, or upload it to D2L, or put it in the TA's mailbox in the main CS office. Handwritten is fine in all cases, or typed…….before class the Dropbox will shutdown at 9am September 14th and there will be no late assignments accepted.

The following are all the questions from the book, you don't have to answer all of them, just the ones above. I tried to get all graphics and tables copied over to this pdf.

1. The two mathematical models of language description are generation and recognition. Describe how each can define the syntax of a programming language.

2. Write EBNF descriptions for the following:
   a. A Java class definition header statement
   b. A Java method call statement
   c. A C **switch** statement
   d. A C **union** definition
   e. C **float** literals

3. Rewrite the BNF of Example_3.4 (Given below) 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>

4. Rewrite the BNF of Example_3.4 (Given in Question #3) to add the `++` and `--` unary operators of Java.

5. Write a BNF description of the Boolean expressions of Java, including the three operators `&&`, `||`, and `!` and the relational expressions.

6. Using the grammar in Example 3.2 (Given Below), show a parse tree and a leftmost derivation for each of the following statements:
   a. `A = A * (B + (C * A))`
   b. `B = C * (A * C + B)`
   c. `A = A * (B + (C))`

Example 3.2 A Grammar for Simple Assignment Statements

<assign> → <id> = <expr>

<id> → A | B | C

<expr> → <id> + <expr>
| <id> * <expr>
| (<expr>)
| <id>

7. Using the grammar in Example 3.4 (Given in Question 3), 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 An Unambiguous Grammar for Expressions

<assign> → <id> = <expr>

<id> → A | B | C

<expr> → <expr> + <term>
| <term>

<term> → <term> * <factor>
| <factor>

<factor> → (<expr>)
| <id>

8. Prove that the following grammar is ambiguous:

<S> → <A>

<A> → <A> + <A> | <id>

<id> → a | b | c

9. Modify the grammar of Example 3.4 (Given in Question 3 or 7) to add a unary minus operator that has higher precedence than either + or *.

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

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?

c. baab
d. bbbab
e. bbaaaaa S
f. 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?

    c. abcd
    d. acccbd
    e. acccbcc
    f. acd
    g. accc

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.

14. Draw parse trees for the sentences aabb and aaaabbbb, as derived from the grammar of Problem_13.

15. Convert the BNF of Example_3.1 (Given Below) to EBNF.

Example 3.1 A Grammar for a Small Language

<program> → **begin** <stmt_list> **end**
<stmt_list> → <stmt>
| <stmt> ; <stmt_list>
<stmt> → <var> = <expression>
<var> → A | B | C
<expression> → <var> + <var>
| <var> - <var>
| <var>

16. Convert the BNF of Example_3.3 to EBNF.

Example 3.3 An Ambiguous Grammar for Simple Assignment Statements

<assign> → <id> = <expr>

<id> → A | B | C

<expr> → <expr> + <expr>

| <expr> * <expr>

| (<expr>)

| <id>

17. Convert the following EBNF to BNF:
- S→A{bA}S→A{bA}
- A→a[b]A