Tutorial 3
Syntax Analysis

1 Syntax Analysis

Question 1.

Given the grammar

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

Show a parse tree and a leftmost derivation for each of the following statements:

a. A = A * (B + C)

b. A = A * B + C

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

Question 2.

Write grammar for the Boolean expressions of Java, including following operators with precedence in descending order and associativity in this table:

Precedence	Operator	Description	Kind
1 (highest)	!	Logical NOT	Unary-Prefix-Right
2	== !=	Relational “equal to” and “not equal to”	Binary-Infix-None
3	< <= > >=	Relational "less than", "less than or equal to", "greater than" and "greater than or equal to"	Binary-Infix-None
4	||	Logical conditional-OR	Binary-Infix-Left
5 (lowest)	&&	Logical conditional-AND	Binary-Infix-Left

Explanation:

- Unary/Binary: Number of operands: one or two
- Prefix/Infix: Position of operator: before or in between its operands
- Right/None/Left: Association

Question 3.

Convert the following EBNF to BNF

```
s   →   a (Ba)*
a   →   A (B)? a
```

Question 4.

a) Prove that the following grammar is ambiguous:

```
s    →   a
a    →   a + a | id
id   →   A | B | C
```

b) Find out what "Left recursion removal" means and perform the left recursion elimination for the above grammar

Question 5.

Find out what "Left factoring" means and perform left factoring for the following grammar

```
stmt   →   IF expr THEN { stmt } ELSE { stmt }
       |   IF expr THEN { stmt }
       |   other
expr   →   TRUE | FALSE
```

Question 6.

Convert the BNF in Question 4 and 5 to EBNF