Assignment #2

BNF

Due on Wed, February 21, 2018

1. Rewrite the BNF of the following example, to give + precedence over * and force + to be right associative

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

2. Rewrite the BNF of the same example in the above question, to add ++ and -- unary operators of Java

3. Using the following grammar,

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

draw parse trees for each of the following statements:

1. A=A*(B+(C*A))
2. B=C*(A*C+B)
3. A=A*(B+(C))

4. Assume the following rules of associativity and precedence for expressions:

Precedence	Highest	*, /, not
		+, -, &, mod
	↑	-(unary)
		=, /= , <, <=, >=, >
		and
	Lowest	or,xor
Associativity	Left to Right	

1) Define BNF grammar that follow above rules, and are able to describe the following expressions. (Assume the only operands are the names: a, b, c, d and e)

- a*b – 1 + c
- a*(b-1)/c mod d
- (a-b)/c & (d * e/a - 3)
- -a or c = d and e
- a>b xor c or d <=17
- -a +b

2) Using the grammar, draw BNF parse trees for these expressions.

3) Rewrite the grammar rules using Extended BNF