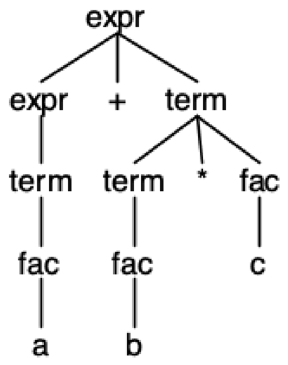
University of Nottingham Malaysia

Compilers Coursework 2

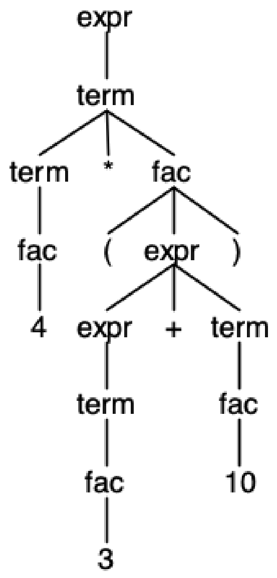
Name: GohXinYee
Student ID: 20093715

Question 1

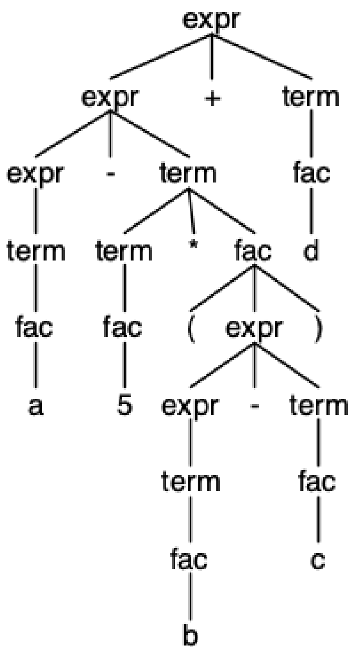
a) a + b ∗ c



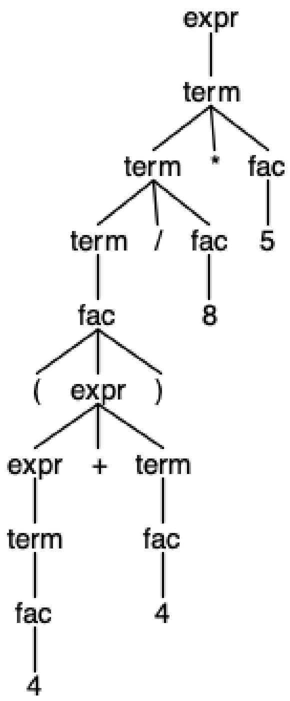
b) 4 ∗ (3 + 10)



c) $a - 5 * (b - c) + d$

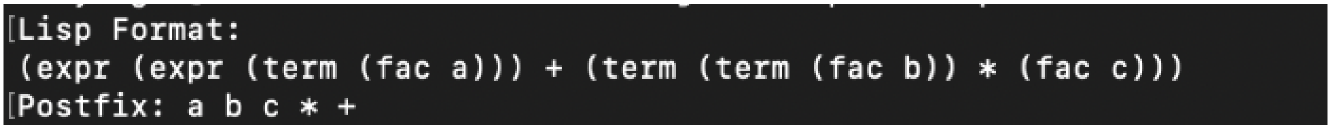


d) $(4 + 4)/8 * 5$



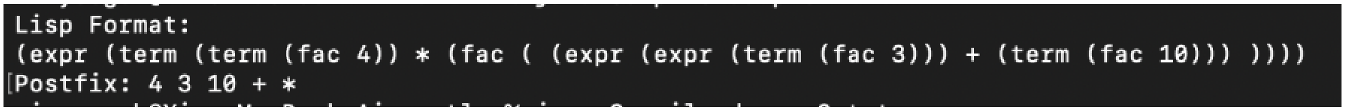
Question 3

a) a + b ∗ c



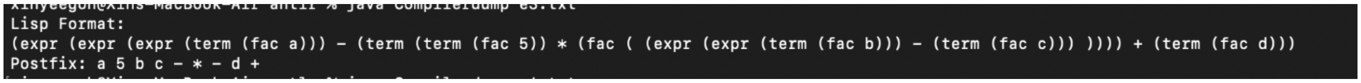
```
[Lisp Format:
 (expr (expr (term (fac a))) + (term (term (fac b)) * (fac c)))
[Postfix: a b c * +
```

b) 4 ∗ (3 + 10)

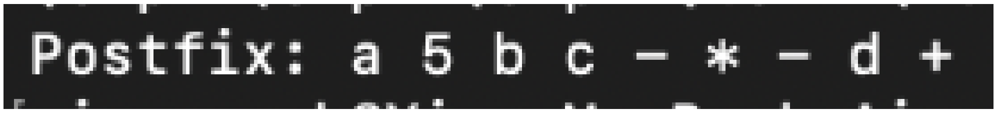


```
Lisp Format:
(expr (term (term (fac 4)) * (fac ( (expr (expr (term (fac 3))) + (term (fac 10))) ))))
[Postfix: 4 3 10 + *
```

c) a − 5 ∗ (b − c) + d

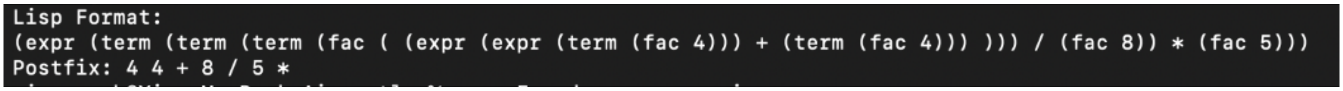


```
xinyeegon@Xins-MacBook-Air antlr % java Compilerdump e3.txt
Lisp Format:
(expr (expr (expr (term (fac a))) - (term (term (fac 5)) * (fac ( (expr (expr (term (fac b))) - (term (fac c))) )))) + (term (fac d)))
Postfix: a 5 b c - * - d +
```



```
Postfix: a 5 b c - * - d +
```

d) (4 + 4)/8 ∗ 5



```
Lisp Format:
(expr (term (term (term (fac ( (expr (expr (term (fac 4))) + (term (fac 4))) ))) / (fac 8)) * (fac 5)))
Postfix: 4 4 + 8 / 5 *
```

Explanation on the grammar Exprdump.g4

The grammar defined is unambiguous. It involves non-terminals *expr*, *term*, and *fac*, terminals +*, -, *, /, (,), id* and *num*, and it ignores whitespace. The start symbol is *expr*.

The order of non-terminals goes from *expr* to *term*, then *fac*. That said, when forming trees, the low precedence expressions (+ -) will be checked for a match before the high precedence expressions (* / (...) and unary minus).

This is so that the high precedence expressions can be formed further down the tree, and thus, be fully evaluated earlier — higher priority — when the tree is visited. This can be seen on the parse trees in **Question 1**.