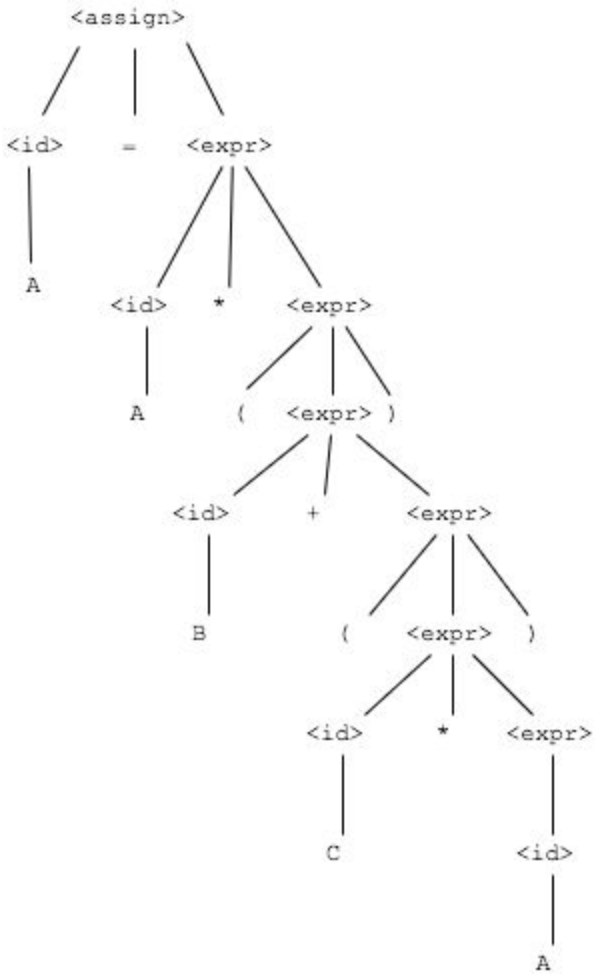
Homework 6
O Max Anderson

Part 2

6. a.

```
A = A * (B + (C * A))
```

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



b.

B = C * (A * C + B)

<assign> → <id> = <expr>

B = <id> * <expr>

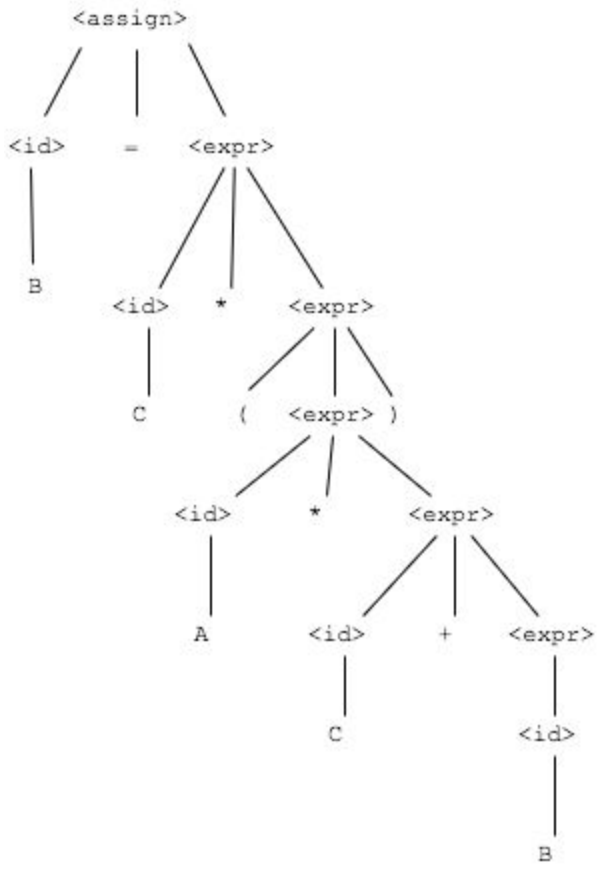
B = C * (<expr>)

B = C * (<id> * <expr>)

B = C * (A * <id> + <expr>)

B = C * (A * C + <id>)

B = C * (A * C + B)



b.

A = A * (B + (C))

<assign> → <id> = <expr>

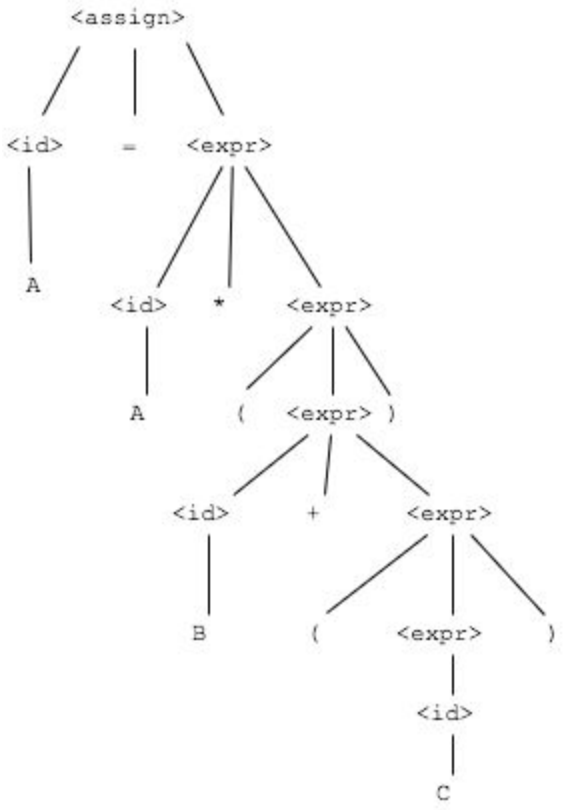
A = <id> * <expr>

A = A * (<expr>)

A = A * (<id> + <expr>)

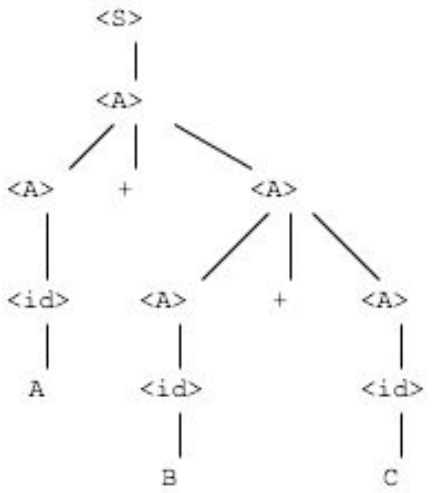
A = A * (B + (<expr>))

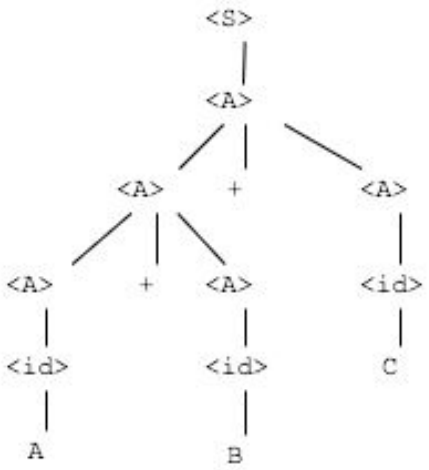
```
A = A * (B + (<id>))
A = A * (B + (C))
```



8.

Two different parse trees for `A + B + C`.





21a.

Java `do-while`

```
do {
      stmt1
      stmt2
} while (cond)
```

Meaning

```
loop: stmt1
      stmt2
      if cond goto loop
out:  ...
```

23a.

```
a = 2 * (b - 1) - 1 {a > 0}

2 * (b - 1) - 1 > 0
2 * (b - 1) > 1
b - 1 > ½
b > 3/2

{b > 1.5}
```

Check

```
b = 2
a = 2 * (2 - 1) - 1
  = 2 * (1) - 1
  = 2 - 1
  = 1
{a > 0} = true

b = 1
a = 2 * (1 - 1) - 1
  = 2 * (0) - 1
  = 0 - 1
  = -1
{a > 0} = false

b = 1.5
a = 2 * (1.5 - 1) - 1
  = 2 * 0.5 - 1
  = 1 - 1
```

```
  = 0
{a > 0} = false
```

25a.

```
if (a == b)
     b = 2 * a + 1
else
     b = 2 * a;
{b > 1}

{B and P} S1 {Q}, {(not B) and P} S2{Q}
---------------------------------------
{P} if B then S1 else S2 {Q}

{(a == b) and P} S1 {b > 1}, {(a != b) and P} b = 2*a {b > 1}
------------------------------------------------------------
{P} if (a == b) b = 2*a + 1 else b = 2*a {b > 1}
```

if case:

```
{a == b and P}
b = 2 * a + 1
{b > 1}
2 * a + 1 > 1
2 * a > 0
a > 0
```

else case:

```
{a != b and P}
b = 2 * a
{b > 1}
2 * a > 1
a > ½
```

We need to strengthen the if case precondition to `a > ½`. The weakest precondition is `a > ½`.

Check:

```
a = 1, should succeed
case (a == b):
     b = 2 * a + 1
       = 2 * 1 + 1
       = 2 + 1
       = 3
```

```
{b > 1} success
case (a != b):
     b = 2 * a
     b = 2 * 1
     b = 2
{b > 1} success

a = 0, should fail
case (a == b):
     b = 2 * a + 1
       = 2 * 0 + 1
       = 1
{b > 1} failure

case (a != b):
     b = 2 * a
       = 2 * 0
       = 0
{b > 1} failure

a = ½, should fail
case (a == b):
     b = 2 * a + 1
       = 2 * ½ + 1
       = 1 + 1
       = 2
{b > 1} success

case (a != b):
     b = 2 * a
       = 2 * ½
       = 1
{b > 1} fail
```

Part 3

M_{if}(<be>, <sl.if>, <sl.else>, s) =>
     if M_{be}(<be>, s)
          M_{sl}(<sl.if>, s)
     else
          M_{sl}(<sl.else>, s)

Reflection

As far as I can tell, every method implemented works as intended. Tested on personal computer and on ceclnx01, compiles and runs properly.

C#

```
------------NonGenericContains------------
True
True
False
True
False
------------Contains------------
True
True
False
True
False
------------IsSorted------------
False
False
True
------------CountIf------------
2
2
0
------------Filter------------
2 4
-2 2 10
abc 34b ABDac

------------TransformIf------------
1 -2 2 -1 10
1 0 2 0 10
abc 34b ABDac
ab 34 AB
```

Python

```
[9, 4, 1, 0, 1, 2, 3]
['hel', 'wor', 'abc']
```