Sami Meharzi

Lab2

2a) (1) $\frac{}{a \in VObjects}$ $\frac{}{b \in VObjects}$

(2) $\frac{s1 \in AObjects \quad s2 \in AObjects}{s1 \,\&\, s2 \in AObjects}$ $\frac{s \in VObjects}{s \in AObjects}$

2b) If we want a result of b & b & b, there is more than one answer:

```
A => A &  A                    A =>  A  &  A
     |    |                          |     |
     A & A & A => A & A & V         A & A & A =>  V & A & A
                 |   |   |                      |    |   |
                 A & A &  b => A & V & b        b & A & A => b & V & A
                               |   |   |                    |    |   |
                               A & b & b => V & b & b       b & b & A => b & b & V
                                            |   |   |                  |   |   |
                                            b & b & b                  b & b & b
```

Since there is not ONE specific way to find an answer, the grammar is defined as ambiguous.

2(c) L(g)::= {a*n , b*n , c*n : n>=0}

2(d) Only sentences 1 and 4 are in the language generated by this grammar.
baab: S => AaBb => baBb => baab
bbaab: S => AaBb => AbaBb => bbaBb => bbaab

2(e) Only sentences 1 and 5 are in the language generated by this grammar.

abcd

```
      S
    /  /\  \
  a  S  c  B
     |     |
     b     d
```

accc

```
      S
    /  /\  \
  a  S  c  B
     |     |
     A     A
     |     |
     c     c
```

3(a)(i) The first grammar is left associative because the recursive non-terminal symbol is on the left of the parse tree.

The 2nd grammar is right associative because the recursive non-terminal symbol is on the right of the parse tree.

(ii) The two grammars do produce the same expression, one by right associative and the other by left associative. The results will be in a combination of: operand operator operand operator ... operand
The difference is determined by if it adds an operand to the left or right

(b) Example: 4 - 1 << 3

```
val exp = 4 - 1 << 3
val minus = (4 - 1) << 3
val shift = 4 - (1 << 3)
if (exp == minus)
    println("- has higher precedence over <<")
if (exp == shift)
    println("<< has higher precendence over -")
```

After using the scala interpreter, we discover the - has higher precedence over <<. By using the expression, 4 - 1 << 3, the result was 24. We compare that answer to (4 - 1) << 3 and 4 - (1 << 3). (4 - 1) << 3 is 24 while 4 - (1 << 3) is -4. Therefore, this shows us that - has higher precedence over <<. This is interesting because of the fact that << is also the same as multiplying by base 2.

(c) frac ::= -n.nIn.nIz.nI-z.nI-n.zIn.zIExpIz.z
Exp ::= z.nEnI-z.nEnIn.nEnI-n.nEnIn.zEnI-n.zEn
z ::= 0
n ::= 1I2I3I4I5I6I...Iinfinity