Problem Set 2 **Due: March 2** **cs2me3 Winter 2004**

Instructions

1. You are encouraged to work in groups of three. If you cannot find two partners, you can work in a group of two. As a last resort, you can work alone.
2. The names of all the partners must be indicated on the assignment.
3. For **Part A** of the assignment, you must submit an electronic copy of your C program via WebCT, and a hard copy of your program should be included when you hand in the entire assignment (Parts A and B) in class.

Part A

1. Write a C program, which takes as input a well-formed algebraic formula, and outputs a Java applet. The applet should display a graphic tree representation of the formula.

   More precisely, write a program `displaytree.c` which compiles under `gcc`:

   ```
   $ gcc -o displaytree displaytree.c
   ```

   and on input `(a*((b+c)*D))`, i.e.,

   ```
   $ displaytree "(a*((b+c)*D))"
   ```

   outputs the following (for example) to the standard output:

   ```
   import java.applet.Applet;
   import java.awt.*;
   public class Coordinates extends Applet {
             public void paint (Graphics page) {
                       setSize (230,140);
                       setBackground (Color.white);
                       page.setColor (Color.blue);
                       page.drawLine (50,20,170,50);
                       page.drawLine (50,20,20,50);
                       page.drawLine (170,50,200,80);
                       page.drawLine (170,50,110,80);
                       page.drawLine (110,80,140,110);
                       page.drawLine (110,80,80,110);
                       page.setColor (Color.lightGray);
                       page.fillOval (40,10,20,20);
                       page.setColor (Color.blue);
                       page.drawOval (40,10,20,20);
                       page.setColor (Color.black);
                       page.drawString ("*",47,24);
                       page.setColor (Color.lightGray);
                       page.fillOval (10,40,20,20);
                                   .
                                   .
                                   .
             }
   }
   ```

Then, the user can "re-direct" the output to a file, that is

```
$ displaytree "(a*((b+c)*D))" > Coordinates.java
```

`Coordinates.java` contains the Java instructions which can now be compiled into Java bytecode as follows:

```
$ javac Coordinates.java
```

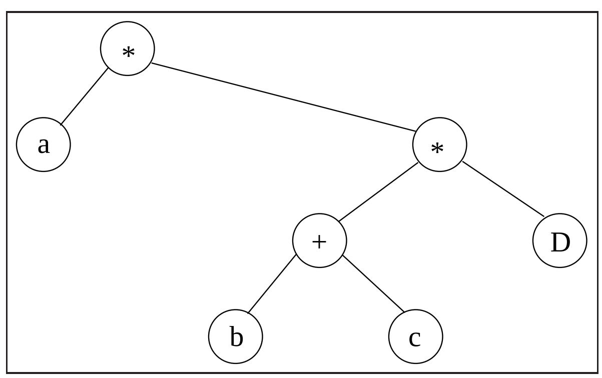
resulting in the applet `Coordinates.class`. Let `Coordinates.html` be the following HTML file:

```
<html>
<head>
<title>Formula Applet</title>
</head>
<body>
<applet code="Coordinates.class" width=100 height=100>
</applet>
</body>
</html>
```

The formula can now be displayed with the command:

```
$ appletviewer Coordinates.html
```

Which displays a figure that looks, for example, like this:



Your program does not have to deal with malformed formulas; it can be assumed that the input formula is a well-formed formula over the alphabet Σ: `{+,*,(,),a-z,A-Z}`. The formulas are defined as follows: `a-z,A-Z` are considered formulas, and if α, β are formulas, then so are $(\alpha + \beta)$ and $(\alpha * \beta)$.

Note that you **do not** have to learn Java for this assignment, as you will only require a handful of graphic drawing commands from Java (i.e., `page.drawLine`, `page.drawOval`, etc.). The main job is to write a program that can parse a given formula, and compute the coordinates of a tree that displays it as nicely as possible.

This is a fun assignment; enjoy it!

Part B

1. Consider the set of all strings of balanced parentheses of two types, round and square. An example of where these strings come from is as follows. If we take expressions in **C**, which use round parentheses for grouping and for arguments of function calls, and use square brackets for array indexes, and drop out everything but the parentheses, we get all strings of balanced parentheses of these two types. For example,

   `f(a[i]*(b[i][j],c[g(x)]),d[i])`

   becomes the balanced-parenthesis string `([]([][][()])[])`. Design a grammar for all and only the strings of round and square parentheses that are balanced.

2. Consider the following grammar:

   $$G = (\{S\}, \{a, b\}, S, \{S \longrightarrow aS|aSbS|\varepsilon\})$$

   (a) Show that G is ambiguous; that is, show that there is a string in $L(G)$ which has two *distinct* parse trees.

   (b) Show that $L(G)$ is the set of all (and only) strings of a's and b's such that every prefix has at least as many a's as b's.

3. Let Σ be an alphabet. A **regular expression** over Σ is defined inductively as follows: any $a \in \Sigma$ is a regular expression, and so are ε and $\emptyset$. If R_1, R_2 are regular expressions, then so are $(R_1), R_1^*, R_1R_2, R_1 + R_2$.

   The semantics of regular expressions are defined inductively as follows: if $a \in \Sigma$, then $L(a) = \{a\}$. Also, $L(\varepsilon) = \{\varepsilon\}$, and $L(\emptyset) = \emptyset$. Furthermore, $L((R_1)) = L(R_1)$, $L(R_1^*) = L(R_1)^*$, $L(R_1R_2) = \{xy|x \in L(R_1) \text{ and } y \in L(R_2)\}$, and $L(R_1 + R_2) = L(R_1) \cup L(R_2)$.

   Show that a language L can be defined by a grammar of type 2 if it can be defined by a regular expression.

   That is, if L is a language ($L \subseteq \Sigma^*$, for some alphabet Σ), then: there exists a grammar G such that $L = L(G)$ if there exists a regular expression R such that $L = L(R)$.

   For example, the language $L = \{\varepsilon, ab, abab, ababab, abababab, \ldots\}$, is given by the regular expression $R = (ab)^*$, and by the type 2 grammar $G = (\{S\}, \{a, b\}, S, \{S \longrightarrow \varepsilon|abS\})$.

4. Prove the Unique Readability Theorem for first order terms. That is, show that every $\mathcal{L}$-term can be uniquely parsed, given the language $\mathcal{L}$.

5. Read the article **"Program Development by Stepwise Refinement"** by Niklaus Wirth, posted on the course's web site (Jan 27 announcement).