CS 211	Computers and Programming Fall 2003 **Prelim II**	11/18/2003

NAME:_____

CU ID:_____

Recitation instructor/time_____

You have one and a half hours to do this exam.

All programs in this exam must be written in Java. Excessively convoluted code will not be graded.

Problem	Score	Grader
1 (5pts)		
2 (20 pts)		
3 (20 pts)		
4 (10 pts)		
5 (15 pts)		
6 (30 pts)		
Total		

1. (Types, 5 points)

   The following definitions and declarations are part of a Java program.

```
interface I1{}
interface I2{}
class C1 implements I1 {}
class C2 implements I2 {}
class C3 extends C1 implements I2 {}
.....
C1 obj1 = ...;
C2 obj2 = ...;
C3 obj3 = ...;
.....
```

   Say whether each of the following assignments is legal or illegal. Consider each assignment by itself. Justify each answer in one or two sentences.

   (a) obj2 = obj1;

   (b) obj3 = obj1;

   (c) obj3 = obj2;

   (d) I1 b = obj3;

   (e) I2 c = obj1;

2. (Class design and inheritance, 20 points)

   Forest Gump has hired you to write a set of Java classes according to the following specifications.

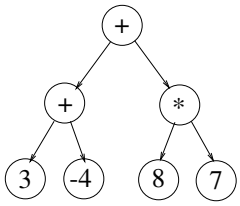

Figure 1: Expression Tree

- Your classes will be used to represent expression trees of the sort shown in Figure 1. These expression trees represent expressions constructed according to the following grammar.

$$E \rightarrow (E + E)$$
$$E \rightarrow (E * E)$$
$$E \rightarrow integer$$

- He wants to build very big trees, so the nodes of your tree should not contain unused fields. In particular, internal nodes of the tree should not have a field that can hold an integer, and leaf nodes should not have a field that can hold an operator. (Hint: The TreeCell class we discussed in lecture is therefore not adequate.)

(a) (10 points) Write down a set of classes that Forest Gump can use to build his trees. Each class must have one or more constructors and the appropriate getter methods, but you do not need to write setter methods.

(b) (10 points) Write a method that takes one of these trees as input and returns the result of evaluating the corresponding expression. For example, the method should return 55 for the tree in Figure 1.

This page left intentionally blank.

3. (Asymptotic complexity, 20 points)

   (a) (7 points) Order the following functions in increasing order of asymptotic complexity: $n!$, 2^n, $nlog(n)$, n, n^2. You do not have to justify your answer.

   (b) (8 points) Java Nagila claims that $2^n = O(3^n)$ and $3^n = O(2^n)$. Is she right? If so, give witness pairs to prove these assertions; otherwise, show that they cannot exist.

   (c) (5 points) Let $f(n)$ and $g(n)$ be positive functions. If $f(n) = O(g(n))$, is $2^{f(n)} = O(2^{g(n)})$? Explain your answer.

This page left intentionally blank.

4. (Graphs, 10 points)

   Answer the following questions with respect to the graph in Figure 2.

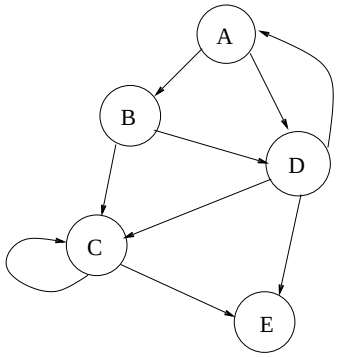

Figure 2: Graph

(a) (2 points) Graph Zeppelin traverses the nodes of this graph in breadth-first order, starting at node A. Write down one sequence of nodes that shows the order in which he might visit the nodes of this graph.

(b) (2 points) Is your answer unique? If not, show a different sequence of nodes that also represents a breadth-first traversal of this graph.

(c) (2 points) Graph Spee traverses the nodes of this graph in depth-first order, starting at A. Write down one sequence of nodes that shows the order in which he might visit the nodes of this graph.

(d) (2 points) Is your answer unique? If not, show a different sequence of nodes that also represents a depth-first traversal of this graph.

(e) (2 points) Is it possible for a sequence of nodes in some graph to represent both a breadth-first and a depth-first visit order? If so, draw such a graph and specify the order. If not, explain why not.

This page left intentionally blank.

5. (Recursion, 15 points)

   A *balanced string of parentheses* is a string that contains only the characters '(' and ')', and satisfies the following condition:

   (a) Either the string is empty, or

   (b) the first character is '(', the last character is ')' and the substring obtained by deleting the first and last characters must itself be a balanced string of parentheses.

   Write a recursive Java class method to check if an input string is a balanced string of parentheses. You can use a helper method if you like.

   You may find the following Java functions useful:

   - s.charAt(i): returns character in position i of String s
   - s.length(): returns length of the String s

This page left intentionally blank.

6. (30 points) Hashley Wilkes wants an implementation of the SearchStructure interface for storing Integer objects. Rather than use a single list, he wants to use an array of 10 lists called *spine* in Figure 3. Integers whose least significant digit is 0 are stored in the list at `spine[0]`, Integers whose least significant digit is 1 are stored in the list at `spine[1]` and so on. Each of the lists is unsorted.

   Write a class called Hashley that implements the SearchStructure interface given below, and uses the data structure of Figure 3. You may assume that only Integers are ever passed to the SearchStructure methods, so no error-checking is necessary. The data structure can have multiple occurrences of an Integer. The `delete` method should remove the first occurrence of its argument from the data structure, if it exists.

   A class for building lists is given at the end of this exam.

   Helpful information:

   - If o is an object of type Integer, the expression (o.intValue() % 10) returns the least significant digit of the integer value stored in o.
   - ```
     interface SearchStructure {
       void insert(Object o); //stick into data structure
       void delete(Object o); //remove first object equal to o from data structure
       boolean search(Object o);//returns true if o is in data structure
       int size();//total number of elements in data structure
     }
     ```
     ```

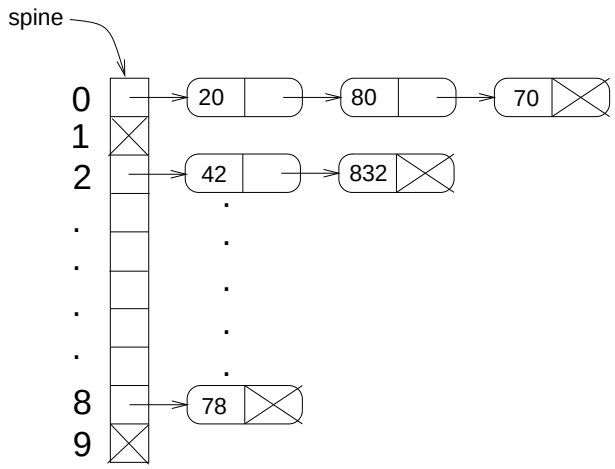

Figure 3: Array of lists for Hashley Wilkes

This page intentionally left blank.

```
class ListCell {

 protected Object datum;
 protected ListCell next;

  public ListCell(Object o, ListCell n){
    datum = o;
    next = n;
  }

  //this is sometimes called the "car" method
  public Object getDatum() {
    return datum;
  }

  //this is sometimes called the "cdr" method
  public ListCell getNext(){
    return next;
  }

  //this is sometimes called the "rplaca" method
  public void setDatum(Object o) {
    datum = o;
  }

  //this is sometimes called the "rplacd" method
  public void setNext(ListCell l){
    next = l;
  }

 public String toString(){
   String rString = datum.toString();
   if (next == null) return rString;
   else return rString + " " + next.toString();
 }
}
```