Computer Science 134
Spring 2000

Final Examination

Question	Points	Score	Description
1	12		Strings
2	15 / 10		Recursion
3	12		Two-dimensional Arrays
4	10		Searching and Sorting
5	8		Program Analysis
6	8		Inheritance
7	8		Streams
8	12		Debugging
total	85		

This examination is closed book. You have 2.5 hours to complete the exam.

Please mark your lecture section:

9am _____ 11am _____

Your Name (Please print) _____

I have neither given nor received aid on this examination

_____

1) Strings

Have you ever come across a word that contains each of the five vowels exactly once? There are quite a few such words, including "authorize", "crematorium", and "Sequoia". Some words even contain all five vowels in order ("facetious"), or in reverse order ("unnoticeably").

The following method is supposed to determine whether a string possesses the amazing property of containing each vowel exactly once. Unfortunately, it doesn't quite work.

(a) Please give the value returned by the method when passed the following strings as parameters:

"facetious"

"authorize"

"ratio"

You may assume that vowels is an array of Strings, containing the values "a", "e", "i", "o", "u".
You may also assume that all strings are lowercase.

```
public boolean facetious(String s)
{
   boolean hasAllOnce;
   for (int i = 0; (i < vowels.length && hasAllOnce); i++)
   {
      int firstOccurs = s.indexOf(vowels[i]);
      if (firstOccurs != -1)
      {
         int secondOccurs = s.indexOf(vowels[i],firstOccurs);
         if (secondOccurs != -1)
         {
            hasAllOnce = false;
         }
      }
      else
      {
         hasAllOnce = false;
      }
   }
   return hasAllOnce;
}
```

(b) Please fix the method so that it works correctly. You should not need to make more than two changes.

2) <u>Recursion</u>

For this question, we would like you to write a recursive method. The good news is that we will let you choose which of two different methods you write. The bad news is that we won't give you as many points for what we think is the easier problem as we will if you solve the harder problem.

The first problem (a) is what we consider the easier problem. A correct solution is worth a maximum of 10 points. A correct solution to the second problem (b) is worth up to 15 points. If you include work on more than one of these problems, please make it very clear which of the two you want us to grade.

(a) An infinite series such as

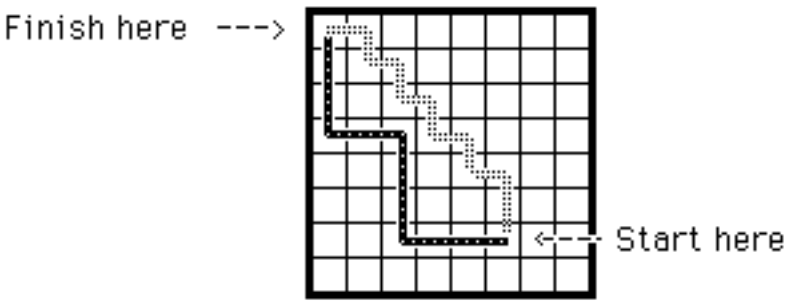
1 - 2 + 3 - 4 + 5 - 6 + . . .

is called an alternating series.

In the case of 1 - 2 + 3 - 4 + 5 - 6 . . . , the sequence of partial sums is 1, -1, 2, -2, 3, -3, . . .

Write a recursive method that calculates the nth partial sum, given a positive integer n.

(b) Suppose that you are given a grid used as a board in a game in which a piece can only be moved to any of its immediate horizontal or vertical neighbors in one move (i.e. to move diagonally requires two steps, one step up or down and one step left or right). Now, consider the problem of finding the shortest sequence of moves that will take a piece from some square in the middle of the grid to the upper left corner of the board. There are obviously lots of paths you might follow that are all just as short as the others. Examples of two such paths are shown in the grid below. Both paths go from the grid position in the 7th row and 6th column to the upper left corner (i.e. the 1st row and 1st column). Each of these paths passes through 12 squares in the grid.



If you have a path in mind and don't feel like counting the moves to see if it is a short as those shown above, all you need to do is make sure that in each step of your path you move either up or to the left. Such paths are always shortest paths.

Write a ***recursive*** method to determine how many such shortest paths there are from a specified square in the grid to the upper left corner. Your method header should look like this:

```
public int countPaths( int startRow, int startColumn )
```

It should return a count of the number of distinct paths consisting only of moves that go up or to the left from the position specified by startRow and startColumn to the upper left corner of the grid.

Note that

```
System.out.println(countPaths(1,1));
```

should output "0". While

```
System.out.println(countPaths(2,2));
```

should output "2".

Solution for 2b:

3) Two-Dimensional Arrays

We have decided to write a program to play the game of Tic-tac-toe. We have decided to store the game as a two-dimensional array of characters. When either X or O takes a turn, we place 'X' or 'O' in the appropriate position in the array.

We would like you to help us out. Specifically, we would like you to implement a method called checkForWin, that checks for a horizontal or vertical win. You need not check for diagonal wins. The method should return true if there is a horizontal or vertical win and should return false otherwise.

The header for the method should be:

```
public boolean checkForWin(int row, int col)
```

Where row and col indicate the position of the most recent move made. You may assume that you have access to the following constants and variable, which represent the current game:

```
// the number of rows and columns in a tic-tac-toe game
private static final int ROWS = 3;
private static final int COLS = 3;

// the game board
private char[][] gameBoard;
```

4) <u>Searching and Sorting</u>

Say that you are asked to write an algorithm that, given an array of integers and a target integer, is to return the number of occurrences of the target in the array.

How many comparisons, **in the worst case**, need to be made if
(a) the array is sorted
(b) the array is not sorted

Please explain your answers.

When discussing search algorithms in class, we analyzed their complexity (that is, how long they would take to run) by counting the number of comparisons the algorithms would make. Why did we only count comparisons? Don't we care about anything else?

5) Program Analysis

What does the following method do? Specifically, assume that words is an array containing the four strings "Space Invaders", "Nibbles", "Pictionary", and "Simon". That is,

```
words [0] = "Space Invaders";
words [1] = "Nibbles";
words [2] = "Pictionary";
words [3] = "Simon";
```

What does the array look like after the method has executed?

```
public void mysteryMethod(String[] words)
{
   for (int i = 1; i < words.length; i++)
   {
      for (int j = 0; j < words.length-1; j++)
      {
           if (words[j].length() > words[j+1].length())
           {
                String temp = words[j];
                words[j] = words[j+1];
                words[j+1] = temp;
           }
      }
   }
}
```

6) <u>Inheritance</u>

Given the class definitions below, what does the code snippet that is at the bottom of the page output?

```
public class Fruit {
   public void squeeze () {
      System.out.println ("Nothing happens");
   }
}


public class Lemon extends Fruit {
   public void squeeze () {
      System.out.println ("Here  comes lemon juice!");
   }
}


public class Banana extends Fruit {
   public void squeeze () {
      System.out.println ("Squoosh!");
   }
}
public class Apple extends Fruit {
}



Fruit [] fruit = new Fruit[3];
fruit[0]  = new Lemon();
fruit[1] = new Banana();
fruit[2] = new Apple();
for (int i = 0; i < fruit.length; i++) {
   fruit[i].squeeze();
}
```

7) Streams

The purpose of the code here is to read a URL and write its contents to a file named "page.html". We provide the code to open and close the streams you will need. You should provide the code to read from the URL and write to the file.

```
import objectdraw.*;
import java.net.*;
import java.io.*;

public class URLSaver extends Controller
{
   public void begin()
   {
      BufferedReader urlIn = null;
      BufferedWriter fileOut = null;

      try {
         URL page = new URL ("http://www.cs.williams.edu/");

         urlIn = new BufferedReader (
                     new InputStreamReader
                           (page.openStream()));
         fileOut = new BufferedWriter (
                     new FileWriter ("page.html"));

         // Add code to read and write the streams









         urlIn.close();
         fileOut.close();
      } catch (MalformedURLException e) {
         System.out.println ("Bad URL");
      } catch (IOException e) {
         System.out.println ("IOException");
      }
   }

}
```

8) Debugging

The purpose of the following program is to display 2 buttons. One button is labeled "Me first!". The other is labeled "Me second!". When the user clicks on either button, the labels on the buttons should swap. The program does not work. When I run it, I get a NullPointerException. Find and fix the NullPointerException. After making that change, do you believe it works? If not, find and fix the remaining error.

```
import objectdraw.*;
import java.awt.*;
import java.awt.event.*;

public class ButtonSwapper extends Controller
                           implements ActionListener
{
   Button button1, button2;

   public void begin()
   {
      setLayout (new FlowLayout());

      Button button1 = new Button ("Me first!");
      add (button1);
      button1.addActionListener (this);

      Button button2 = new Button ("Me second!");
      add (button2);
      button2.addActionListener (this);
   }

   public void actionPerformed (ActionEvent evt) {
      // Swap the labels on the buttons
      button1.setLabel(button2.getLabel());
      button2.setLabel(button1.getLabel());
   }

}
```