**Computer Science 134**
Spring 2003

Final Examination

| Question | Points | Score | Description |
|---|---|---|---|
| 1 | 12 | | Strings |
| 2 | 10 | | Concurrency |
| 3 | 10 | | Exceptions |
| 4 | 12 | | Searching and Sorting |
| 5 | 12 | | Arrays |
| 6 | 20 | | Multidimensional Arrays |
| 7 | 8 | | Inheritance |
| 8 | 16 | | Recursion |
| Total | 100 | | |

This examination is closed book. You have 2.5 hours to complete the exam. **Good luck!**

Please mark your lecture section:

9am _____ 10am _____

Please mark your lab section:

Monday _____ Tuesday _____

Your Name (Please print) _____

I have neither given nor received aid on this examination

_____

**1) Strings (12 points)**

You are to write a method `getParenStrings` of a class `A` that takes a `String` as its only parameter and returns a `String` containing all parenthesized substrings of the given `String`. The returned `String` should contain all of the parenthesized substrings (without the parentheses), separated by new lines.

For example, the code

```
A a = new A();
String sentence = "Some words (like these) are in parens (you see)";
System.out.println(a.getParenStrings(sentence));
```

should print out

```
like these
you see
```

Recall that the Java `String` class includes a method `indexOf(String s, int i)` that will give the position within the `String` of the first occurrence of the substring `s`, starting at position `i`, and a method `substring(int startPos, int beyondEndPos)` that returns the substring of the given `String`.

You may assume that there are no nested parentheses, and that each open parenthesis has a matching close parenthesis later in the string.

Please write the method below:

```
public class A {

    public String getParenStrings(String str) {









    }
}
```

**2) Concurrency (10 points)**

In your first array laboratory, you wrote the Simon program, which used a `SongPlayer` class. To help detect secret messages in songs (a favorite of conspiracy theorists), we now wish to write a `ReverseSongPlayer` class. The idea is that this song player should be an `ActiveObject` that takes a song and plays it backwards forever. The class also has a method `changeSong` that erases the old song in the array (wouldn't want the bad guys to be able to find it!) and creates a new song array. Here is the code of the class:

```
import java.applet.*;
import objectdraw.*;

public class ReverseSongPlayer extends ActiveObject {
      AudioClip[] song;
      int size, current;

      public ReverseSongPlayer(AudioClip[] startSong, int size) {
         song = startSong;
         this.size = size;
         start();
      }

      public void run(){
         while (true) {
            playNotes();
         }
      }

      public void playNotes() {
         for (int note = size -1; note >= 0; note--){
               song[note].play();
               pause(250);
            }
      }

      public void changeSong(AudioClip[] newSong, int newSize) {
         for (int note = 0; note < size; note++) {
            song[note] = null;
         }
         song = newSong;
         size = newSize;
      }
}
```

There is a `Controller` class that creates the `ReverseSongPlayer` and also calls `changeSong` whenever a user pushes a `Button` in the window that pops up when the program begins.

a. Everything works fine except occasionally when the button is pushed and the `Controller` calls method `changeSong` when the `ReverseSongPlayer` is busy playing the song. Please explain what unfortunate thing might happen in this circumstance.

b. How could the programmer prevent this bad thing from happening when the user pushes the button when a song is playing. You are not allowed to change any of the code inside any of the method bodies in `ReverseSongPlayer` (*e.g.*, you are not allowed to omit the `for` loop in `changeSong`).

**3) Exceptions (10 points)**

You have all had the experience of getting `NullPointerException`s when running programs in Java. A frustrated programmer kept getting these exceptions when executing the following line of code:

```
x = fst.getValue() + snd.getValue() / third.getValue();
```

where `x` is an `int` variable and each of the calls to `getValue()` also returns an `int`. Out of frustration, the programmer replaced that line with the following:

```
try{
  x = fst.getValue() + snd.getValue() / third.getValue();
catch (NullPointerException exc) {
  System.out.println("fst is: "+fst+", snd is: "+snd+",
                     third is: "+third);
  x = 0;
}
System.out.println(x);
```

You may assume that if `fst`, `snd`, and `third` are different from `null` then their values would print the results of sending `getValue()` to the object referred to by the variable. [It is not the case that all of the following scenarios will raise null pointer exceptions.]

a. Please discuss what is printed in the revised code when `fst.getValue()` returns 3, `snd.getValue()` returns 6, and `third` is `null`.

b. Please discuss what is printed in the revised code when `fst.getValue()` returns 3, `snd.getValue()` returns 6, and `third.getValue()` returns 2.

c. Please discuss what is printed in the revised code when `fst.getValue()` returns 3, `snd.getValue()` returns 6, and `third.getValue()` returns 0. Hint: a division by 0 throws an `ArithmeticException`.

**4) <u>Searching and Sorting</u> (12 points)**

For parts a-c, assume that you are given an array of 1,024 ($=2^{10}$) strings, each of which contains a single word. In each case, you should use the most efficient of the algorithms we discussed that is applicable.

a. If the array is not sorted, how many of the strings will you need to look at in the worst case to determine whether or not the array contains a given word?

b. Now, suppose the array of strings has been sorted alphabetically. How many words will you need to look at in the worst case to determine whether or not the array contains a given word?

c. If your starting array is not sorted, you may choose to sort the array alphabetically before searching for the word. Is this worthwhile if you are searching for just one word in the array? What about if you are searching for over 1,000 words in the array? Explain briefly.

d. Recall that selection sort works by first finding the largest element in the array and moving it to the last position in the array. It then finishes by sorting the rest of the array. Suppose that sorting an array of 1,000 elements takes 2 seconds. Approximately how long will it take to sort 10,000 elements using selection sort?

**5) Arrays (12 points)**

A computer science student, we'll call him John Doe, probably at some other college far, far away from here, was working on his lab program. The class John was working on is supposed to keep a collection of `int` values in an array. The `insert` method places a new value at position 0 in the array, after moving up existing values up to make room. The `remove` method returns the value at the end of the array, and removes it from the array.

John's program isn't working. Here is his code:

```
public class IntCollection {

    private static final int ARRAY_SIZE = 5;
    private int[] intArray;
    private int currentSize;

    public IntCollection() {

        intArray = new int[ARRAY_SIZE];
        currentSize = 0;
    }

    public void insert(int newValue) {

        for (int index=0; index<currentSize; index++) {
            intArray[index+1] = intArray[index];
        }
        intArray[0] = newValue;
        currentSize++;
    }

    public int remove() {

        currentSize--;
        return intArray[currentSize];
    }
}
```

John has been testing his `IntCollection` with the following code segment:

```
        IntCollection ic = new IntCollection();

        ic.insert(17);
        ic.insert(23);
        ic.insert(3);
        ic.insert(8);
        System.out.println(ic.remove());
        System.out.println(ic.remove());
        System.out.println(ic.remove());
        System.out.println(ic.remove());
```

He expects the program to print

```
17
23
3
8
```

a. What does the program print?

b. Correct the code on the previous page so that it behaves correctly.

c. What will happen if your corrected `IntCollection` class is used as follows?

```
IntCollection ic = new IntCollection();

for (int i=0; i<10; i++) {
    ic.insert(i);
}
```

d. How should you modify the `insert` method avoid the problem demonstrated in part c?

## 6) Multidimensional Arrays (20 points)

Design a class `GameBoard` that has as an instance variable, `gameArray`, that is a two dimensional array of `Color`s. The constructor should take parameters `numRows`, `numCols`, and `startColor`, where `numRows` is the number of rows in the array, `numCols` is the number of columns, and `startColor` is a value of type `Color`. The constructor should create an array of the appropriate size and place `startColor` in every slot of the array.

The class will also contain a variety of other methods (*e.g.*, to change colors in rows and columns), but the only one you need to write is a method `checkWin` that takes parameters `rowNum` and `colNum` and returns `true` only if both parameters are legal (*i.e.*, `rowNum` is a legal row number and `colNum` is a legal column number), and all the colors in the row given by `rowNum` are the same color and all of those in the column given by `colNum` are the same color. If any of these fail to hold then the method should return `false`.
*Be sure to write both the constructor and checkWin method.*

**7) <u>Inheritance (8 Points)</u>**

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

```
public class Road {
   public int speedLimit () {
      return 55;
   }
}


public class CityStreet extends Road {
   public int speedLimit () {
      return 30;
   }
}


public class Interstate extends Road {
   public int speedLimit () {
      return 65;
   }
}
public class StateHighway extends Road {
}


Road [] road = new Road[3];
road[0]  = new CityStreet();
road[1] = new Interstate();
road[2] = new StateHighway();
for (int i = 0; i < road.length; i++) {
   System.out.println(road[i].speedLimit());
}
```

**8) Recursion (16 Points)**

One of our earliest examples of recursion was the Broccoli problem. On this exam, we want you to add a method that simulates the effect of a worm burrowing into a part of the broccoli and cutting off the flow of nutrients to all parts of the broccoli that extend from that part. Thus you need a method called `damage` that takes a `Location` as a parameter. If that `Location` is on a part of the broccoli then that part should be turned black as well as all other parts extending from there, up to and including the flowers. You need to add appropriate code for this method to the `BroccoliPart` interface and the classes `BroccoliBranch` and `Flower`. The relevant parts of the original code are given below. Please write your answers on a separate page (and indicate in which class each new piece of code belongs).

```
public interface BroccoliPart {
  // amount to move the broccoli part
  public void move(double x, double y);

  // whether a point is contained in the broccoli part
  public boolean contains(Location point);
}
```

---

```
public class BroccoliBranch implements BroccoliPart {
 …

  // stem of broccoli
  private AngLine stem;

  // branches of broccoli
  private BroccoliPart left, center, right;

  // Draw broccoli by recursively drawing branches (and flower)
public BroccoliBranch(Location startLocation, double size,
                      double direction, DrawingCanvas canvas)
  {
    // Draw stem and color green
    stem = new AngLine(startLocation, size, direction, canvas);
    stem.setColor(BROCCOLI_COLOR);
   …
    if ( size > MINSIZE ) {                  // draw branches
      left = new BroccoliBranch(destLocation, size * TOP_FRACT,
                                direction + Math.PI/9.0, canvas);
      …
    } else {                                 // draw flowers
      left = new Flower(destLocation, size * TOP_FRACT,
                        direction + Math.PI/9.0, canvas);
      …
    }
  }
```

```
  // @param x,y amount to move broccoli
  public void move( double x, double y) {
    stem.move(x,y);                                  // move stem

    left.move(x,y);                                  // move other parts
    center.move(x,y);
    right.move(x,y);
  }

  // @param point location to be checked for containment in broccoli
public boolean contains( Location point ) {
    return (stem.contains(point) || left.contains(point) ||
            center.contains(point) || right.contains(point));
  }

}
```

---

```
public class Flower implements BroccoliPart {
  …
  private AngLine stem;       // stem of broccoli
  private FilledOval bud;           // flower of broccoli plant

  // Draw flower
  public Flower(Location startCoords, double size,
                double direction, DrawingCanvas canvas) {
    stem = new AngLine(startCoords, size, direction, canvas);
    stem.setColor(BROCCOLI_COLOR);
    …
    bud = new FilledOval(destLocation,BUD_SIZE,BUD_SIZE,canvas);
    bud.setColor(Color.yellow);
  }

public void move(double x, double y) {
    // move stem
    stem.move(x,y);

    // move bud
    bud.move(x,y);
  }

public boolean contains (Location point) {
    return stem.contains(point) || bud.contains(point);
  }


}
```

**Answer to Question 8:**