Prelim 1, Questions

CS 2110, 27 September 2018, 7:30 PM

	1	2	3	4	5	6	Total
Question	Name	Short answer	Exception handling	Recursion	OO	Loop invariants	
Max	1	34	8	16	31	10	100
Score							
Grader							

The exam is closed book and closed notes. Do not begin until instructed. This handout contains the questions; you will answer the questions on the accompanying answer handout.

You have **90 minutes**. Good luck!

Write your name and Cornell **NetID**, **legibly**, at the top of **every** page of the answer handout! There are 6 questions on 5 numbered pages, front and back. Check that you have all the pages. When you hand in your exam, make sure your pages are still stapled together. If not, please use our stapler to reattach all your pages!

You can use the back of the last page of this question handout as scrap paper. We have scrap paper available. If you do a lot of crossing out and rewriting, you might want to write code on scrap paper first and then copy it to the exam so that we can make sense of what you handed in.

Ambiguous answers will be considered incorrect. Your answers should fit easily into the space provided.

In some places, we have abbreviated or condensed code to reduce the number of pages that must be printed for the exam. In others, code has been obfuscated to make the problem more difficult. This does not mean that it's good style.

1. Name (1 point)

Write your name and NetID, **legibly**, at the top of **every** page of the answer handoout.

2. Short Answer (34 points)

(a) 6 points. This question appears fully on the answer sheet.

(b) 5 points. This question appears fully on the answer sheet.

(c) 7 points. Use classes `Instrument` and `Drum` below to answer the question on the answer sheet.

```
public abstract class Instrument {
    public Instrument() {
        System.out.println("I am an instrument!");
    }

    public void makeSound() {
        System.out.println("curses!");
    }
}

public class Drum extends Instrument {
    public Drum() {
        super();
        System.out.println("I am a drum");
    }

    public void makeSound() {
        System.out.println("BOOM!");
    }
}
```

(d) 6 points. Implement function `differ` whose specification is given on the answer sheet. Here are examples:

```
char[] array1=  {'b', 'r', 'a', 'c', 'y'};
char[] array2=  {'r', 'b', 'a', 'c', 'y'};
differ("bracy", array1) is false
differ("Bracy", array1) is true
differ("bracyyy", array1) is true
differ("bracy", array2) is true
```

(e) 6 points. On the answer handout, write 4 JUnit test cases for `differ` as specified on the answer sheet. If you forget the correct syntax, simply write the arguments that you would test. For each one, BRIEFLY state why the case is unique and significant.

(f) 4 points. This question appears fully on the answer sheet.

3. Exception handling (8 Points)

On the answer handout, write the output to the console for the three statements `oof(-1,-1);`, `oof(0,0);`, and `oof(1,1);`. Procedure `oof` is given below.

```
public static void oof(int x, int y) {
    System.out.println("A");
    int res= x / (y * 10);
    try {
        if (y == 0 || y == 1)
            throw new IllegalArgumentException();
        System.out.println("C");
        res= 1 / (y+1);
        System.out.println("D");
    }
    catch (ArithmeticException e) {
        System.out.println("E");
        if (y == -1)
            throw new RuntimeException();
        System.out.println("F");
    }
    catch (Exception e) {
        System.out.println("G");
    }
    System.out.println("H");
}
```

4. Recursion (16 Points)

(a) 8 points Recursive function `sevenUp` is given below Execute the three calls shown on the answer handout and, for each, write down on the answer handout what is printed. But on each, you can stop after executing 4 println statements.

```
public static void sevenUp(int n) {
  if (n > 0) {
      System.out.println(n);
      if (n != 7) {
           if (n % 2 == 0) // n is even, add 2 and halve it
               sevenUp((n + 2) / 2);
           else // n is odd, multiply by 3 and subtract 3
               sevenUp(3*n - 3);
      }
  }
}
```

(b) 8 points The answer handout contains a recursive procedure `spacify`; complete its method body. You can use function `f`, given below. **You must use recursion; do not use a loop!**

```
/** = m but with leading 0's, if necessary, to have at least 3 chars.
      Precondition: 0 <= m */
public static String f(int m ) {
   if (m >= 1000) return "" + m;
   if (m >= 100) return "0" + m;
   return "00" + m;
}
```

5. Object-Oriented Programming (31 points)

Below is abstract class `Student`, which is used throughout this problem:

```
public abstract class Student                    {
   private String name; // Name of student

   /** Constructor: a student with name n.  */
   public Student(String n) {name= n;}

   /** Return true if student passed all courses, false otherwise */
   public boolean passed() {... Assume implemented ...}

   /** Return true only if student took CS2110 and passed all courses */
   public abstract boolean canGraduate();

   public abstract void graduate();

   /** Return a negative int, 0, or positive int depending on whether
     * this students's name comes before, is the same as, or comes after
     * ob's name, in dictionary order. Throw a ClassCastException
     * if ob is not a Student.  */
   public int compareTo(Object ob) {
      ...
   }

}
```

(a) 5 points Recall that interface `Comparable` declares abstract function `compareTo(Object ob)`. This function is declared in class `Student`. On the answer sheet, complete its body and explain one other necessary change to class `Student` so that it implements interface `Comparable`.

Two hints are given on the answer handout.

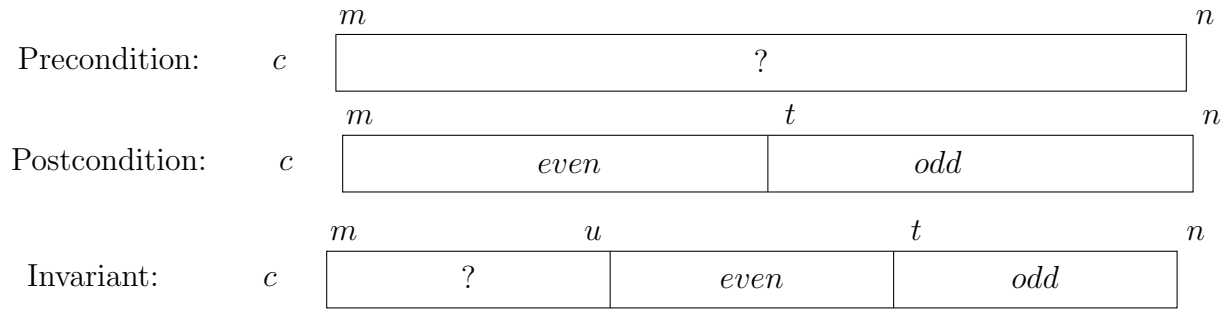
(b) 12 pts Class `Undergrad` on the answer handout extends `Student`. On the answer sheet,
(1) Complete the body of the constructor.
(2) Complete the return statement in function `canGraduate`, assuming the boolean variable is assigned properly (you do not have to do this).
(3) Complete the body of procedure `graduate`.

(c) 5 points On the answer sheet, complete the body of function `toString`, which is to be added to class `Undergrad`. The result should contain the name of the student and the number of courses. You could just return those two values with a space between them.

(d) 9 points Class `CSstudent` (on the answer sheet) is a subclass of `Undergrad`. Each method has a syntax error, so it doesn't compile. On the answer sheet, identify the three syntax errors.

6. Loop Invariants (10 points)

Below are the precondition, postcondition, and invariant of a loop that swaps the odd values in $c[m..n-1]$ to the end of $c[m..n-1]$. Note that m is not necessarily 0, n is not necessarily $c.length$, and m and n should not be changed. You do not have to be concerned with declaring variables.



On the answer sheet, complete questions (a), (b), and (c).