Prelim 1

CS 2110, 13 March 2018, 5:30 PM

	1	2	3	4	5	6	Total
Question	Name	Short answer	Exception handling	Recursion	OO	Loop invariants	
Max	1	30	11	14	30	14	100
Score							
Grader							

The exam is closed book and closed notes. Do not begin until instructed.

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

Write your name and Cornell **NetID**, **legibly**, at the top of **every** page! There are 6 questions on 10 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!

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.

Write your answers in the space provided. Ambiguous answers will be considered incorrect. You should be able to fit your answers 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.

Academic Integrity Statement: I pledge that I have neither given nor received any unauthorized aid on this exam. I will not talk about the exam with anyone in this course who has not yet taken prelim 1.

(signature)

1. Name (1 point)

Write your name and NetID, **legibly**, at the top of **every** page of this exam.

2. Short Answer (30 points)

(a) 6 points. Below are six expressions. To the right of each, write its value.

1. `(int) 'g' == 'g'`
2. `(char)('z' - 2)`
3. `new Character('c') == new Character('c')`
4. `(new Double(2.0 * 3.0)).equals(new Double(1.0 * 6.0))`
5. `"I love CS2110".substring(5).substring(2, 5)`
6. `k == 2 || 6 / (k - 2) != 7` (Note: k is of type int)

(b) 5 points. State whether each of the following statements is true or false.

1. `String x= "Hello World"; x= '3';` will compile.
2. The following code declares a two-dimensional array of ints: `int[2] myInts;`
3. "Generic types" refers to the built-in Java types: byte, short, int, long, float, double, boolean, char.
4. It is possible for a method to be both overridden and overloaded.
5. One purpose of wrapper class Integer is to treat an int value as an object.

(c) 4 points.

Consider the following classes:

```
public class Boat1 {
  public abstract String getName();
}
public class Boat2 extends Boat1 {
  public Boat1 makeFriend() {
    Boat1 friend= new Boat1();
    return friend;
  }
}
public class Boat3 extends Boat1 {
  public String getName() {
    return "Boaty McBoatface";
  }
}
public class Boat4 extends Boat1 {
  private String name;
  public Boat4(String n) {
    name= n;
  }
}
```

I. Which classes need to be abstract in order to compile?

II. Which classes will fail to compile even if they are abstract?

(d) 8 points.

This function returns the sum of all the even integers in an array:

```
/** Return the sum of the even integers in array b (return 0 if b is null). */
public static int sumE(int[] b) {
  if (b == null) return 0;
  int sum= 0;
  for (int i= 0; i < b.length; i= i + 1) {
    if (b[i] % 2 == 0){
      sum= sum + b[i];
    }
  }
  return sum;
}
```

I. Write down, in English, at least 5 distinct test cases that you need to consider.

II. Write down, in Java, the code for these test cases. Assume that function sumE is declared in class U. Hint: to declare an array with the values 1, 2, and 3, you can use: new int[]{1, 2, 3}.

```
@Test
public void testSumE() {




}
```

(e) 4 points.

Write the algorithm for executing the procedure call `p(3, 2.1)`.

(f) 3 points.

In A3, the doubly-linked list class DLLList had these fields:

```
private Node first; // first node of linked list (null if size is 0)\\

private Node last;  // last node of linked list (null if size is 0)\\

private int size;   // Number of values in the linked list.
```

while inner class Node had these fields:

```
    private Node prev; // Previous node on list (null if this is first node)\\

    private E val;     // The value of this element\\

    private Node next; // Next node on list. (null if this is last node)
```

Change the class invariants above so that the doubly linked list is a circular doubly linked list.

3. Exception handling (11 Points)

(a) 8 points. What-input-is-needed-to-get-output. Using the class and procedure below, answer the questions to the right, providing an appropriate procedure call as needed. Write “none” if no procedure call will give the desired output.

```
public class R {
  public static void b(int k, String s) {
    int x= 0;
    int y= 0;
    try {
       System.out.println("A");
       y= s.length();
       System.out.println("B");
       x= k / y;
       System.out.println("C");
    } catch(NullPointerException npe) {
       System.out.println("D");
       x= k / (k-1);
       System.out.println("E");
    } catch(RuntimeException re) {
       System.out.println("F");
       try {
          int z= k / (y-2);
          System.out.println("G");
       } catch(RuntimeException r) {
          System.out.println("H");
       }
    }
    System.out.println("I");
    int z= k / (k-5);
    System.out.println("J");
  }
}
```

2 points per option (all-or-nothing)

Give one call of procedure b that will print the following:

```
A
D
E
I
J
```

What call on b does not print "J"?

What call on b prints this:

```
A
B
C
I
J
```

What call on b will result in a thrown exception?

(b) 3 points. Executing a try-statement. Write the algorithm (in English) for executing the following `try-catch` block, which is not within another try-block.

`try { S1 } catch (RuntimeException re) { S2 } .`

4. Recursion (14 Points)

(a) 6 points

Execute the three calls griesSeq(1); griesSeq(5); and griesSeq(8); and write the return value of the calls in the places provided below.

```
public static int griesSeq(int n) {
  if (n == 0 || n == 1) return 1;
  if (n % 2 == 0) {
    return griesSeq(n - 1);
  }
  return n * griesSeq(n - 1);
}
```

Return value for `griesSeq(1)`:
Return value for `griesSeq(5)`:
Return value for `griesSeq(8)`:

(b) 8 points Consider the following class representing Elephants. Write the body of recursive procedure `numNames`. **You must use recursion; do not use a loop!**

```
public class Elephant {
    private String name; // not null
    private Elephant child; // null if this Elephant has no child

    /** Constructor: an instance with name n and child c.
      * Precondition: n is not null. */
    public Elephant(String n, Elephant c) {
        name= n;
        child= c;
    }

    /** Return the number of Elephants in this Elephant's family that have
      * name n. This Elephant's family consists of this Elephant, its child,
      * its child's child, its child's child's child, etc.  */
    public int numNames(String n) {








    }
}
```

5. Object-Oriented Programming (30 points)

Below is class `Course`, which you will be using throughout this problem:

```
public class Course {
  public int number;
  public String name;

  /** Constructor: Course with number num and name n. */
  public Course(int num, String n) {
    number= num; name= n;
  }
}
```

(a) 4 points Complete the body of the constructor in class `Student`:

```
public class Student {
  private int studentID;
  private double gpa;
  private String name;


  /** Constructor: Student with name n, student ID id, and gpa gpa. */
   public Student(String n, int id, double gpa) {





  /** Return true if this student has a gpa > 2.0. */
  public boolean inGoodStanding() {
    return gpa > 2.0;
  }
}
```

(b) 10 points Complete the body of the constructor and function `inGoodStanding` in class `EnrolledStudent` on the next page.

```
   /** A Student who is currently enrolled in an institute. */
public class EnrolledStudent extends Student                               {
  private int maxC;           // Max number of courses to take at one time.
  private Course[] schedule;  // Student is taking courses
  private int numC;           // schedule[0..numC-1]

   /** Constructor: Newly enrolled Student taking 0 courses, with
   *  name n, student ID id, gpa gpa. Can take at most max
   *  courses at one time. */
  public EnrolledStudent(String n, int id, double gpa, int max) {




  }

   /** Return true if this student has a gpa > 2.0 and is taking
     * at least 3 courses. */
  public @Override boolean inGoodStanding() {


  }











}
```

(c) 16 points Below is a declaration of interface `Enrolled`. Above, do whatever is necessary in class `EnrolledStudent` to have it implement `Enrolled`.

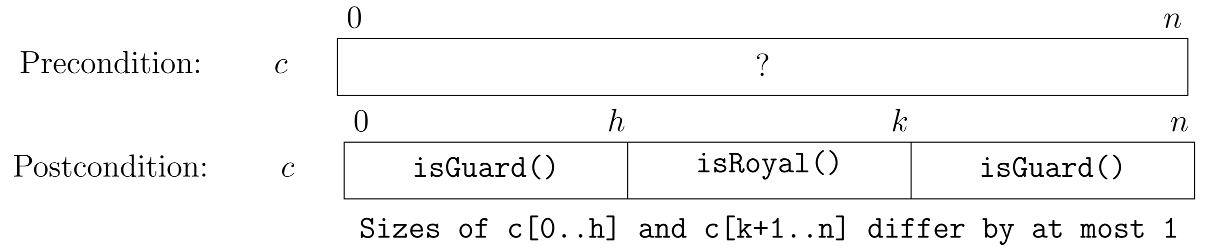
```
public interface Enrolled {
  /** If student is taking the max number of courses allowed, return false.
    * Otherwise, add c to the student's schedule and return true. */
  public boolean addCourse(Course c);

  /** Return the enrolled student's schedule. */
  public Course[] getSchedule();
}
```

6. Loop Invariants (14 points)

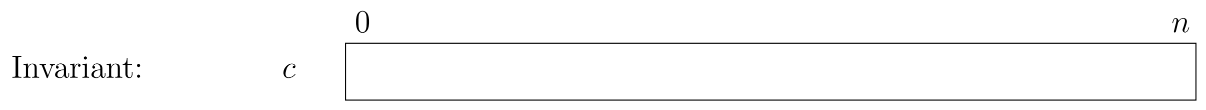
Consider class `Person` below, which has two methods `isRoyal()` and `isGuard()` to determine whether a `Person` is a member of the British Royal Family or the British Guard, respectively. You will use the four loopy questions to develop a single loop (with initialization) that modifies an array c to surround the Royal Family with British Guards.

```
class Person {
   private boolean guard;
   public Person (boolean g) { guard= g; }
   public boolean isGuard() { return guard; }
   public boolean isRoyal() { return !guard; }
}
```

(a) 6 points Consider this precondition and postcondition for an array c of `Person` objects.



Complete the invariant below to generalize the above array diagrams. You will have to introduce a new variable. Place your variables carefully; ambiguous answers will be considered incorrect. Note: Several different invariants can be drawn; draw any one of them.



(b) 1 point Below, write the initialization that truthifies the invariant:

(c) 2 points Write a while-loop condition and write something in the loop body that makes progress toward termination. (*Hint: When the loop condition is false, the invariant must imply the postcondition.*)

(d) 5 points Write a loop body that keeps the invariant true. (*Hint: use procedure* `swap(c,i,j)` *to swap array elements* $c[i]$ *and* $c[j]$.)

```
initialization:

while (                                      ) {




}
```