Prelim 1, Solution

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'` true char is numerical type
2. `(char)('z' - 2)` 'x' char is a numerical type
3. `new Character('c') == new Character('c')` false These are two separate objects.
4. `(new Double(2.0 * 3.0)).equals(new Double(1.0 * 6.0))` true
5. `"I love CS2110".substring(5).substring(2, 5)` "CS2"
6. `k == 2 || 6 / (k - 2) != 7` (Note: k is of type int) true

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

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

(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?

Boat1 must be abstract because it contains an abstract method. Assume it has been made abstract.

Boat2 fails to compile even if it is made abstract because it contains "new Boat1()".

Boat3 compiles. It overrides abstract method getName in Boat1.

Boat4 must be abstract because it extends Boat1 but doesn't override method getName.

(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.

Solution: they must hit 5 of the categories below (multiple tests from one category count only as 1.)
0. sumE(null) is 0
1. sumE([]) is 0
2. sumE([some odd integers]) is 0
3. sumE([some even integers]) is sum of even integers
4. sumE([some even integers and odd integers]) is sum of even integers
5. negative evens count as evens, negative odds don't.

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() {

assertEquals(0, U.sumE(null));
assertEquals(0, U.sumE(new int[0]));
assertEquals(12, U.sumE(new int[]{2, 4, 6}));
assertEquals(0, U.sumE(new int[]{1, 3, 5}));
assertEquals(6, U.sumE(new int[]{1, 2, 3, 4, 5}));
assertEquals(-2, U.sumE(new int[]{-1, -2}));









}
```

(e) 4 points.

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

1. Push a frame for the call onto the call stack. (It contains space for the parameters and all local variables.)
2. Assign the argument values to the parameters (in the frame for the call).
3. Execute the method body (using the frame for the call for the local vars and pars).
4. Pop the frame for the call from the call stack. (There is no need to talk about putting the retuned value on the stack because p is a procedure.)

(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.
The most important point is to change the definitions of prev and next. We change them to:

Previous node on list (pointer to node last if this is node first)
Next node on list (pointer to node first if this is node last)

Now it is truly a circular doubly linked list. We did not look at whether you changed the defs of first and last because they were not that important point. However, one can quibble with the current definitions of first and last, because there is no longer the concept of a "first" node and a "last" node. One can change this in several ways.

1. One could say that the so-called "first" node could be changed at any time to any node, since the list is circular.
2. One could change the def of first to "any node of the list" and change last to say it contained first.prev (null if first is null).
3. One could change the def of first to "any node of the list" and delete field last completely! One needs only one pointer into the list.

3. Exception handling (11 Points)

(a) 8 points. What-input-is-needed-to-get-output. Using the given class and procedure, 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
```

b(0, null);

What call on b does not print `"J"`?

b(1, null);

What call on b prints this:

```
A
B
C
I
J
```

b(1, "x");

What call on b will result in a thrown exception?

b(1, null);

(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 }` .

Execute S1. If S1 throws a RuntimeException, execute S2. If S1 throws any other exception, throw it out to the call of the method that contains this try-statement. (The previous sentence is necessary. Without it, one does not know what happens if S1 throws any other exception.)

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)`: 1
Return value for `griesSeq(5)`: 15
Return value for `griesSeq(8)`: 105

(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) {


         int k= name.equals(n) ? 1 : 0;
         return child == null ? k : k + child.numNames(n);
    OR
         if (child == null) { return name.equals(n) ? 1 : 0; }
         return (name.equals(n) ? 1 : 0) + child.numNames(n);
    YOU CAN ALSO WRITE IT WITHOUT USING A CONDITIONAL EXPRESSION


    }
}
```

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) {
     name= n;
     studentID= id;
     this.gpa= gpa; // "this." is necessary
  }

  /** 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`:

```
/** A Student who is currently enrolled in an institute. */
public class EnrolledStudent
             extends Student implements Enrolled {
  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) {
    super(n, id, gpa);
    maxC= max;
    schedule= new Course[maxC]; //new Course[max] also works.
    numC= 0;
  }

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

  /** 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) {
    if (numC >= maxC) { // numC == maxC is OK
      return false;
    }
    schedule[numC]= c;
    numC= numC + 1;
    return true;
  }

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

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

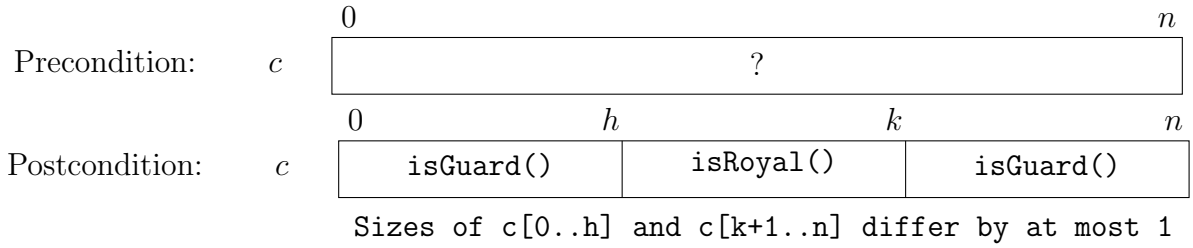
```
public interface Enrolled {
  /** If student is taking the max number of courses allowed, return false.
    * Otherwise, add c to the student's schedule and return true. */
  public boolean addCourse(Course c);

  /** 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.

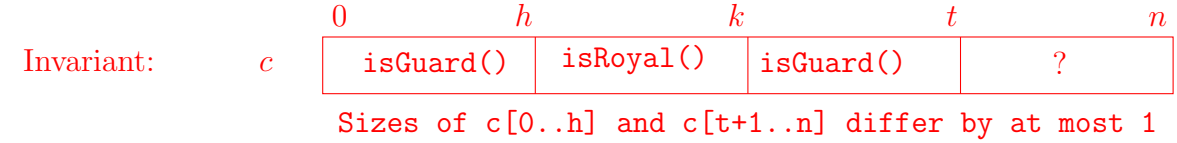
Sample Answer: This problem is similar to the Dutch National Flag except that the first and last sections have the same property (elements are guards) and it additionally requires that sizes of guard sections differ by at most 1.

Many did not copy the part of the invariant that the sizes of the two guard sections differ by at most 1. Some complained, saying that the question did not tell them to do this. But think this way. One loopy question asks you to prove this:

invariant AND falsity of loop condition IMPLY postcondition.

How can you prove this if that part of the postcondition is not in the invariant?

Some drew a variable right above a vertical line, creating ambiguity. Please don't do that! Below, note how 0 is to the right of a line and all others to the left of a line.



Note that there are several other correct invariants; alternative correct solutions might have the ? section in a different place or might have the new index t on the other side of the line (or called something else). A different invariant will result in different answers to 5(b), (c), and (d).

(b) 1 point 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: make sure that when the loop condition is false, the invariant implies 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* $c[j]$.)

It is wrong to use variables in the loop (except for local variables of the repetend) that are not defined in the invariant.

This solution comes from a student. The ? section is at the right but it still requires at most one swap per iteration! The interesting point about the repetend is the case when (at the start) c[t+1] is a guard and a guard should be put into the left section c[0..h]. Instead of swapping c[t+1] into that section, guard c[k+1] is put into it!

```
h= -1; k= -1; t= -1;
while (t < n) {
     t= t+1;
     if (c[t].isRoyal()) { k= k+1; swap(c, k, t); }
     else if (h+1 < n-t) { h= h+1; k= k+1; swap(c, k, h); }
     }
}
```

The attempt given below has two successive if-statements each of which processes a ? value c[t]. The is incorrect. Suppose the ? section has ONE value in it, a Royal. The first if-statement removes it from the ? section, putting it in the Royal section. Thus, the ? section now has 0 values, and any attempt to process a value in it is erroneous —it may even result in an array-index-out-of-bounds exception. Thus, an else is needed, as shown above.

```
h= -1; k= -1; t= -1;
while (t < n) {
     t= t+1;
     if (c[t].isRoyal()) { k= k+1; swap(c, k, t); }
     if (c[t].isGuard()  &&. h+1 < n-t) { h= h+1; k= k+1; swap(c, k, h); }
     }
}
```