Prelim 1. Solution

CS 2110, 28 September 2017, 7:30 PM

	1	2	3	4	5	6	Total
Question	Name	Short answer	Exception handling	Recursion	OO	Loop invariants	
Max	1	31	12	16	30	10	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 8 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 (31 points.)

(a) 5 points. Below are five expressions. To the right of each, write its value.

1. `(int) 'a' == (char) 'a';` true
2. `new Double(4.3) == new Double(4.3)` false Objects are different
3. `(new Integer(2+1)).equals((Object)(new Integer(3)))` true auto unboxing / equals
4. `'a' >= "acd".charAt(1) ? "yes" : "no"` "no"
5. `"oddsAndEnds".substring(3).substring(1, 2)` "A"

(b) 5 points. For each of the following statements, state whether it is true or false.

1. In Java, it is illegal to make a class a private. false.
2. If the first statement of a constructor you write is not a call on another constructor, a default constructor call in inserted. true
3. Objects of wrapper class Character are immutable, so class Character has no procedures for changing the character wrapped in a Character object. true
4. Local variables in a method maintain their value inside of recursive calls to that method. false
5. The expression `i < 3` is evaluated exactly 4 times during execution of this loop: true

```
for (int i= 0; i < 3; i= i + 2)  i= i - 1;
```

(c) 4 points. Write the algorithm for evaluating this new-expression: new CC((1+3, 2).
1. Create (draw) an object of class CC, with default values in fields.
2. Execute the constructor call CC((1+3, 2).
3. Yield as value of the new-expression the name of (pointer to) the new object.

(d) 6 points. Implement method `countDiffs`, below. Do not use recursion.

```
   /** Return the number of elements of b that are different from
  * the preceding element.
  * Example: countDiffs([1, 0, 0, 1])  is  2.
  * countDiffs([1, 2, 1, 1, 1, 1, 3])   is   3. */
public static int countDiffs(int[] b) {
   int val= 0;
   // inv: val = no. of elems in b[0..i-1] different from preceding elem
   for (int i= 1; i < b.length; i= i+1) {
       if (b[i-1] != b[i]) val= val + 1;
   }
   return val;
}
```

(e) 6 points

Consider the declarations given below. In the righthand column, for each of the 6 sections of code, write whether it produces no error, a runtime error, or a compile-time error. Assume that each section is independent of the others.

Hint: It helps to draw objects.

```
interface I1 {};
interface I2 {};

class A0 {};
class A1 implements I1 {};
class A2 implements I2 {};
class A3 extends A0 implements I1 {};
class A4 extends A1 implements I2 {};
```

```
(1) I2 x4= (A4)(new A1());

(2) A0 x1= new A3();

(3) I1 x3= new A4();

(4) I2 x6= new A2();
    I2 x7= new A3();
    x6= x7;

(5) I1 x5= (A1)(new A4());

(6) A2 x2= new I2();
```

Answers to e:
(1) run-time error (2) no error
(3) no error (4) compile-time error
(5) no error (6) compile-time error

(f) 5 points.

Consider the class definitions given below. In the righthand column, for each of the 6 sections of code, write whether it executes the method defined in C1, executes the method defined in C2, produces a compile-time error, or produces a runtime error. Assume that each section is independent of the others.

Hint: It helps to draw objects.

```
public abstract class C1 {
    public abstract void f1();
    public void f2() {...};
    public void f3() {...};
}

public class C2 extends C1 {
    public void f1() {...};
    public void f3() {...};
    public void f4() {...};
}
```

```
(1) C1 c= new C1();
    c.f1();

(2) C1 c= new C1();
    c.f2();

(3) C1 c= (C1) new C2();
    c.f1();

(4) C1 c= (C1) new C2();
    c.f2();

(5) C1 c= (C1) new C2();
    c.f3();
```

Answers to f
(1) compile-time error (2) compile-time error
(3) method in C2 (4) method in C1
(5) method in C2

3. Exception handling (12 Points)

Execute the three calls E.m(-2, true); E.m(2, true); E.m(2, false); on procedure m shown below and write the output of the calls to println in the places provided on the right. If executing a call to E.m does not result in any output, write "none".

```
public class E {
  public static void m(int i, boolean b) {
    System.out.println("1");
    int x= 1 / (i+2);
    try {
      System.out.println("2");
      if (b)
        throw new RuntimeException();
      System.out.println("3");
      x= 6 / (2 - i);
      System.out.println("4");
    } catch (RuntimeException e) {
      System.out.println("5");
    } catch (Exception e) {
      System.out.println("6");
      if (b) throw new RuntimeException();
      System.out.println("7");
    }
    System.out.println("8");
  }
}
```

Console for `E.m(-2, true)`:
1
Console for `E.m(2, true)`:
1
2
5
8
Console for `E.m(2, false)`:
1
2
3
5
8

4. Recursion (16 Points)

(a) 8 points

To the right is a declaration of a node of a singly linked list. Example: Using [3, 5] to represent a linked list with two nodes p1 and p2, we have p1.value = 3, p1.next = p2, p2.value = 5, and p2.next = null.

```
public class Node {
    public int value;
    public Node next;
}
```

Write procedure change, below. Use of a loop will give you 0 points.

```
    /** n is (a pointer to) a node of a singly linked list. It may be null.
   * Change the value in n to the value in n.next (if n.next is not null)
   * and do the same for the rest of the nodes in the linked list.
   * For example, calling change on the list [3, 6, 8, 4] should change
   * the list to [6, 8, 4, 4]. */
public static void change(Node n) {
    if (n == null || n.next == null) return;
    n.value= n.next.value;
    change(n.next);
}
```

(b) 8 points

Write the body of the following function. Use of a loop or class String gives you 0 points.

```
  /** Return the number of 0's in the decimal representation of n.
   * Example: If n = 0, return 1. If n = 5030, return 2.
   * Precondition: 0 <= n. */
 public static int sumZeros(int n) {
     if (n == 0) return 1;
     if (n < 10) return 0;
     return sumZeros(n/10) + (n%10 == 0 ? 1 : 0);
  }
```

5. Object-Oriented Programming (30 points)

This question deals with new drivers taking their written road test and having them graded. Below is the definition of `NewDriver`, which you will be using throughout this problem:

```
public class NewDriver {
  private String name;
  private String address;

  /** Constructor: an instance with unknown name and address. */
  public NewDriver() {}

  /** Constructor: An instance with name n and address an. */
  public NewDriver(String n, String an) {
    name= n;  address= an;
  }
}
```

(a) 10 points Below are two class declarations. Complete the bodies of their constructors and function `questNumb` in class `AnswerKey`. Be careful; pay attention to access modifiers.

```
/** Instance is a new driver's
  * answers to a road test. */
public class RoadTest {
  private boolean[] answers;
  private NewDriver driver;

  /** Constructor: instance with
   *  answers ans and driver d.
   * Prec: ans is not null. */
  public RoadTest(boolean[] ans,
                  NewDriver d) {
      answers= ans;
      driver= d;
   }

  /** = the answers. */
  public boolean[] answers()
      {return answers;}

  /** = the new Driver. */
  public NewDriver driver()
      {return driver;}

  /** = answer to question q. */
  public boolean getAnswer (int q) {
       return answers[q];
  }
}
```

```
/** Instance gives answers to a RoadTest
  * and the max score on the RoadTest. */
public class AnswerKey extends RoadTest {
    int max; // max score on road test


  /** Constructor: instance with answers
   *  ans and null driver. max score is m.
   * Prec: ans.length > 0.  */
  public AnswerKey(boolean[] ans, int m) {
     super(ans, null);
     max= m;
   }


  /** Return number of correct answers
    * in ans; sol gives the solution.
    * the grade is the percent right * max. */
  public int grade(RoadTest ans) {...}


  /** Return the number of
   *  questions in this roadTest */
  public int questNumb() {
     return answers().length;
   }
}
```

(b) 8 points Below are function equals to go in classes RoadTest and AnswerKey. Write their bodies to implement their specifications.

```
   /** This goes in class RoadTest.
    * Return true iff this and ob are
    * of the same class and their
    * drivers are the same object. */
public @Override boolean
                    equals(Object ob) {
  if (ob == null  ||
        getClass() != ob.getClass())
     return false;
  RoadTest rt= (RoadTest)ob;
  return driver == rt.driver;
}
```

```
   /** This goes in class AnswerKey.
    * Return true iff this and ob are
    * of the same class, their drivers
    * are the same object, and their
    * max scores are equal. */
public @Override boolean
                      equals(Object ob) {
   if (!super.equals(ob))
      return false;
   AnswerKey ak= (AnswerKey) ob;
   return max == ak.max;
}
```

(c) 6 points Below is a declaration of class Clerk. Complete its constructor.

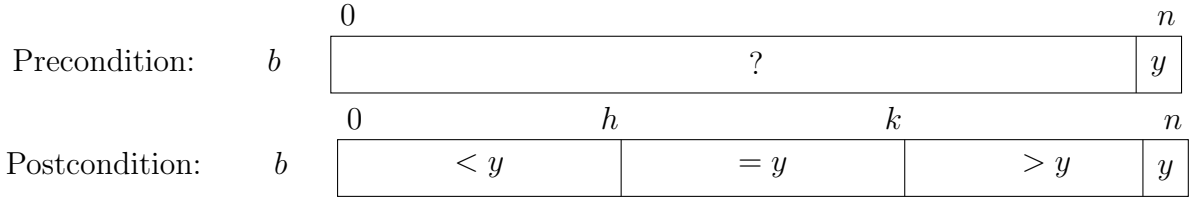
```
   /** An instance is a clerk who grades road tests for a new drivers. */
public class Clerk   implements ClerkName  {
    private String name;     // Clerk's name
    private NewDriver[] d;   // new drivers to grade
    private AnswerKey ansk;  // answer key to road test

    /** Constructor: Clerk with name name, grading road
      * tests d, with answer key ansk. */
    public Clerk(String name, NewDriver[] d, AnswerKey ansk) {
        this.name= name;
        this.d= d;
        this.ansk= ansk;
     }

    /** Return clerk administering road tests.*/
    public String clerkName() {
       return name;
    }
}
```

(d) 6 points Below is a declaration of interface ClerkName. Above, do whatever is necessary in class Clerk to have it implement ClerkName.

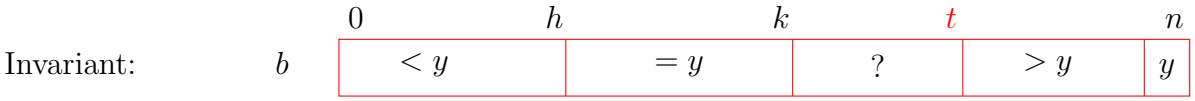
```
public interface ClerkName {
    /** Return the clerk administering road tests. */
    String clerkName();
}
```

6. Loop Invariants (10 points)

(a) 6 points Consider the following precondition, invariant, and postcondition for array b.



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) 4 points Write down the four loopy questions to be answered in proving that a loop is correct.

1. How does it start —what initialization makes the invariant true?
2. When can it stop —what condition together with the invariant makes the result true?
3. How does the repetend make progress toward termination?
4. How does the repetend keep the invariant true?