Prelim 1

CS 2110, March 15, 2016, 5:30 PM

	0	1	2	3	4	5	Total
Question	Name	True False	Short Answer	Object-Oriented	Recursion	Loop Invariants	
Max	1	20	14	25	19	21	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** at the top of **every** page! There are 5 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 its good style.

Academic Integrity Statement: I pledge that I have neither given nor received any unauthorized aid on this exam.

_____

(signature)

0. Name (1 point)

Ensure that your name and NetID is written on **every** page of this exam.

1. True / False (20 points)

Circle T or F in the table below.

a)	T	F	The only difference between an abstract class and a non-abstract class is that an abstract class can have abstract methods.
b)	T	F	The expression `'a' + 1 == 'b'` evaluates to `true`.
c)	T	F	`int[] x= new int[0]` causes an error at run time because we cannot instantiate (create) an array of length 0.
d)	T	F	If `var` is a public field in class `A`, and class `B` extends `A`, then for a variable `x` of type `B`, the expression `x.var` is legal.
e)	T	F	The assignment `Object obj= new char[4];` is legal.
f)	T	F	If class `Dog` extends class `Animal`, and variable `Animal pet` contains a pointer to a `Dog` object, then the statement `Dog spike= pet;` is legal.
g)	T	F	A class can directly extend at most one superclass and can directly implement an arbitrary number of interfaces.
h)	T	F	Let `b` be a non-null `int[]`. The expression `b[b.length]` always throws an `ArrayIndexOutOfBoundsException`.
i)	T	F	Suppose class `Bar` implements interface `Foo`. The statement `Bar bar= new Foo();` is legal.
j)	T	F	Suppose `W` is an abstract class. The declaration `W v;` is legal.
k)	T	F	Suppose `a` is a variable of type `String`. Execution of the expression `"Prefix" + a` will modify the object referenced by `a`.
l)	T	F	The Java expression `Integer.valueOf(1234) == Integer.valueOf(1234)` always evaluates to `true`.
m)	T	F	Interfaces can have a non-static fields.
n)	T	F	If `b` and `c` are objects of class `A` and class `A` has not overridden `equals`, then `b.equals(c)` will evaluate to `true` if and only if the values of every field in `b` and `c` are equal.
o)	T	F	Suppose `x` and `y` are variables of type `Object`. If `x` is `null`, then `x.equals(y)` will evaluate to `true` if and only if `y` is `null`.
p)	T	F	The default value for a field of type `Integer` is `0`.
q)	T	F	Execution of the statements `String s1= "java"; String s2= s1.replace('v', 'V');` makes `s1` reference the string `"jaVa"`.
r)	T	F	If an **abstract** class implements an interface, it does not have to provide an implementation for every method in that interface.
s)	T	F	The statement `throw new A()` is valid if and only if `A` is a non-abstract class that directly or indirectly extends `Throwable`.
t)	T	F	Within a constructor, we can call a constructor of the super class using `super(...)` anywhere in the body of the constructor.

2. Short Answer (14 points)

(a) 8 points Below is a method `m`. Answer the following questions.

(i) (2 points) What is the result of `m("2")`?

(ii) (2 points) What is the result of `m("3a")`?

(iii) (2 points) What is the result of `m("5")`?

(iv) (2 points) What is an example of an argument to `m` that would result in `-1`?

```
public static int m(String number) {
    String[] arrB= new String[]{"5", "5", "1", "10", "99", "125", "2"};
    int x= 0;
    try {
        x= Integer.parseInt(number);
        String value= arrB[x+1];
        int target= Integer.parseInt(value);
        return (40 + target) / (5 - x);
    } catch (NullPointerException e) {
        return -3;
    } catch (NumberFormatException e) {
        return 0;
    } catch (ArithmeticException e) {
        String defaultEntry= arrB[7];
        return Integer.parseInt(defaultEntry);
    } catch (ArrayIndexOutOfBoundsException e) {
        return -1;
    } catch (Exception e) {
        return -2;
    }
}
```

(b) 3 points Consider the following application:

```
public class Prelim {
    private static int i;
    private int j;

    public Prelim(int a, int b) {
        i = a;
        j = b;
    }

    public String toString() {
        return i + " " + j;
    }

    public static void main(String[]args) {
        Prelim p1 = new Prelim(0, 1);
        Prelim p2 = new Prelim(2, 3);
        System.out.println(p1 + " " + p2);
    }
}
```

What does running class `Prelim` as an application print to the console?

(c) 3 points Consider interface `SomeInterface` and class `SomeClass` as defined below:

```
public interface SomeInterface {
    void someFunction();
    void someGuess();
}

public class SomeClass implements SomeInterface {
    public void someGuess() {
        System.out.println("When in doubt, choose C");
    }
}
```

Are these definitions valid? Explain.

3. Object-Oriented Programming (25 points)

(a) 3 points You are hired by RolePlayGames Inc, and you need to design the next game taking place in a fantasy world. You are in charge of designing a class of wizardry students that are part of FireGate Wizardry School. In this school, they specialize in fire spells. Your characters, for practice, are making duels.

Your first job is to design class `FireGateStudent`. Complete the body of method `launchSpell`, below. You do not need to assert preconditions.

```
public class FireGateStudent {
    private int magicShield; //magic shield of the student. >= 0
    private String name;     //name of the student. Is not null

    /** Constructor: instance with name (!= null) and shield (>= 0) */
    public FireGateStudent(String name, int shield) {
        this.name= name;
        magicShield= shield;
    }

    /** Return the student's name **/
    public String getName() { return name; }

    /** Return true iff this student has any magic shield remaining. **/
    public boolean isShielded() { return magicShield > 0; }

    /** If this student is shielded, launch a fire spell at opponent. To do that:
      * Reduce the opponent's shield by 10, but don't let it get below 0.
      * Precondition: opponent is not null. **/
    public void launchSpell(FireGateStudent opponent) {
        // FILL THIS IN









    }
}
```

(b) 3 points A new teammate is working on a second Wizardry school, named BlizzardHall. In this school, students study ice spells instead of fire spells. But BlizzardHall students' spells only halve magic shields rather than strictly decrease them. Here is your teammate's design for these students:

```
public class BlizzardHallStudent {
    private int magicShield; //magic shield of the student. >= 0
    private String name; //Name of the student. Is not null

    /** Constructor: instance with name (!= null) and shield (>= 0) */
    public BlizzardHallStudent(String name, int shield) {
        this.name= name;
        magicShield= shield;
    }

    /** Return the student's name **/
    public String getName() { return name; }

    /** Return true iff this student has any magic shield remaining. **/
    public boolean isShielded() { return magicShield > 0; }

    /** If this student is shielded, launch a blizzard spell at opponent.
      * To do that: Divide the opponent's shield by 2, rounding down.
      * Precondition: opponent is not null. **/
    public void launchSpell(BlizzardHallStudent opponent) {
        // FILL THIS IN









    }
}
```

(c) 19 points Your boss just told you that, in the inter-school duel games, some students from FireGate might have to duel against students from BlizzardHall. She has asked you to write a class `Duel` with a method `duel` that performs a duel and returns the name of the winner. But now you realize a limitation of you and your teammate's design: `FireGateStudent`s can `launchSpell`s only at other `FireGateStudent`s, not at `BlizzardHallStudent`s, and vice versa. Introduce a new class `Student` below that addresses the design problem, and mark up the earlier `FireGateStudent` and `BlizzardHallStudent` with the changes necessary to take advantage of that new class. Then fill in the code for `duel` below using the new class so that `duel` works for both `FireGateStudent`s and `BlizzardHallStudent`s. You can omit comments.

You will be graded on the quality of your design and the accuracy of your code. Any mark up that appears to be intentionally ambiguous will be considered incorrect. Also, recall that keyword `protected` makes a member visible to all subclasses.

```
public class Duel {
    /** Perform a duel between two students.
      * They take turns launching a spell at each other, starting with s1.
      * The duel stops when either s1 or s2 is no longer shielded.
      * Return the name of the student that is still shielded.
      * Precondition: s1 and s2 are not null and are shielded */
    public static String duel(                  s1,                    s2) {









    }
}
```

4. Recursion (19 Points)

(a) 10 points We want to compute the sum of the values in a linked list `ll` using a call like `sumNode(ll.getHead())`. Complete function `sumNode` below according to its specification. Use recursion. Do **not** use a loop. We include only those parts of the classes that you need. We omit comments in the classes; everything is similar to A3 except that it is a singly linked list.

```
public class LinkedList {
    private Node head= null;

    public Node getHead() {
        return head;
    }
}
```

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

```
/** Return the sum of the values of node and its
  * successors. If node is null, return 0. */
public static int sumNode(Node node) {









}
```

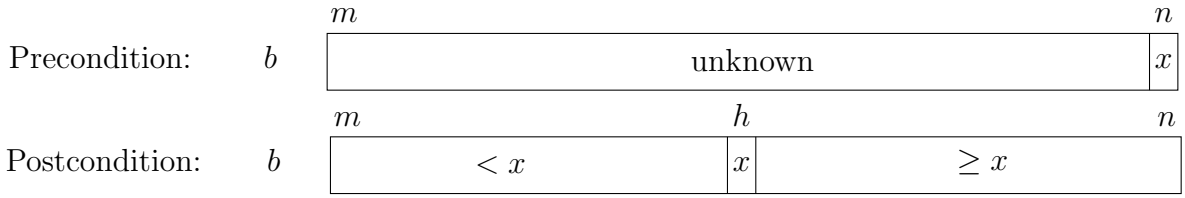
(b) 9 points Write the body of function `sep`, given below. Do not use a loop; use recursion. Do not change parameter `n` into a String and then use String operations like s.length() to figure out where to put blanks. Use int operations / and %. You may use function `to3`, given below.

```
/**  Return n as a string but with spaces every three digits
  *  (this convention is used in many European countries).
  *  Precondition: n >= 0.
  *  Example: for n = 43, return  "43"
  *  Example: for n = 1435, return  "1 435"
  *  Example: for n = 5000123567, return  "5 000 123 567" */
public static String sep(int n) {



}
```

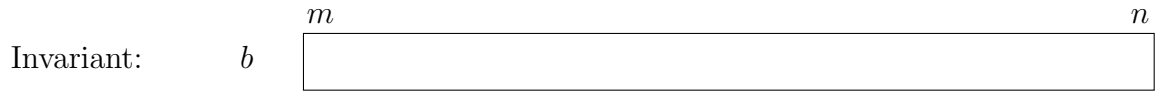
```
/** Return n as a string, with 0's prepended if necessary
   * to make it at least 3 characters. Precondition: n >= 0 */
public static String to3(int n) {
    return n < 10 ? "00" + n : (n < 100 ? "0" + n : "" + n);
}
```

5. Loop Invariants (21 points)

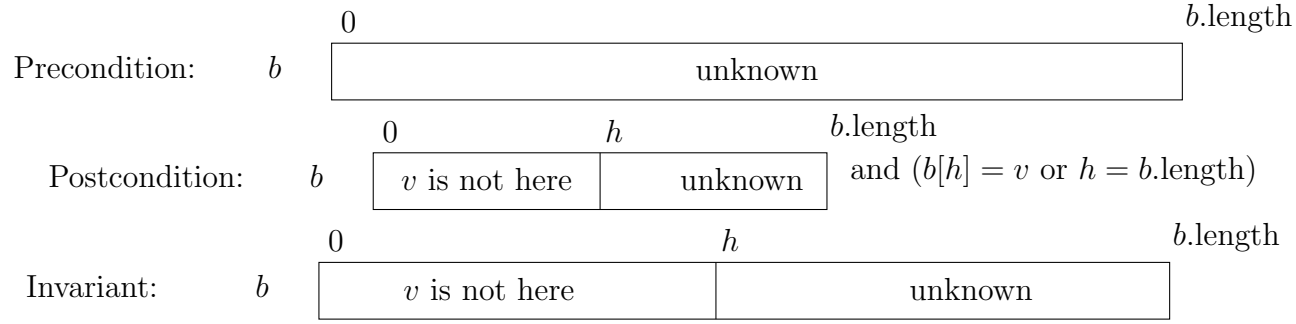
(a) 6 points Consider the following precondition and postcondition.



Generalize the above array diagrams, completing the invariant below. Your generalization can introduce a new variable. Be sure to place your variables carefully; ambiguous answers will be considered incorrect.



(b) 8 points Consider the following precondition, postcondition, and invariant.



Complete function `linearSearch` below according to its specification using the precondition, postcondition, and invariant above. You will be graded on how well you follow the invariant and use the four loopy questions.

```
/** Return the index of the first occurrence of v in b.
  * If v is not in b, return b.length
  * Precondition: b is not null */
public static int linearSearch(int[] b, int v) {




    return h;
}
```

(c) 7 points Consider the following loop with initialization:

```
//Store the product of m..n in z
//Precondition Q: m <= n
int z= 1;
int k= m;
//invariant P: z = product of m..k and m <= k <= n
while (k <= n) {
    z= z * k;
    k= k + 1;
}
//Postcondition R: z = product of m..n
```

Is the above loop invariant correct? If you think the loop invariant is correct, you must explain why *every* loopy question is satisfied. If you think the loop invariant is incorrect, you must pick *one* loopy question and explain why it is not satisfied.