Prelim 1 Solutions

CS 2110, March 10, 2015, 5:30 PM

	1	2	3	4	5	Total
Question	True False	Short Answer	Recursion	Object Oriented	Loop Invariants	
Max	20	15	20	25	20	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 11 numbered pages, front and back. Check that you have all the pages. When you hand in your exam, make sure your booklet is 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.

1. True / False (20 points)

a)	T	F	Finding the largest value in a sorted array with n values takes constant time.
b)	T	F	Inserting a value into the middle of an array of length n takes $O(n)$ time.
c)	T	F	If `a` and `b` hold pointers to objects of class A and `equals` is not overridden in class A, `a.equals(b)` will return true only if every field in `a` and `b` are equal.
d)	T	F	The default value for a field of type `Boolean` is `null`.
e)	T	F	Every Java class except `Object` extends exactly one other Java class.
f)	T	F	If a non-abstract class implements an interface, it must provide an implementation for every method in the interface.
g)	T	F	The declaration `Character ch = 'a';` is not valid, as the `new` operator must be used.
h)	T	F	If you do not put an access modifier on a method declared in an interface, you can access that method only from classes in the same package.
i)	T	F	`Object c = new char[50];` is a legal statement in Java and will not cause a `RuntimeException`.
j)	T	F	If `W` is an interface, the declaration `W e;` is illegal.
k)	T	F	Execution of the statement `Dog[] values = new Dog[3];` calls the constructor for Dog three times to initialize `values[0]`, `values[1]` and `values[2]`.
l)	T	F	If a variable in some class is declared static there will be just a single copy of that variable.
m)	T	F	In each class `C` (say), Java always puts the constructor `public C(){}`
n)	T	F	Within a constructor, we can call another constructor of the same class using `super`.
o)	T	F	If class A has two methods with signatures `m(int)` and `m(String)`, we say that `m` is overloaded.
p)	T	F	Execution of `int x = 54321; byte c = x;` will truncate 54321 to an 8-bit number and store it in `c`.
q)	T	F	If `s` and `t` are variables of type `String`, expression `s + t` creates a new String object.
r)	T	F	Sorting a linked list allows us to use binary search instead of linear search on the list.
s)	T	F	For the statement `throw new C();` to be valid, C must be a subclass of `Exception`
t)	T	F	Assume `class B extends A` and `A x= new B();`. Then `x.m()` is valid if and only if `m` is declared in class A or one of its superclasses.

2. Short Answer (15 points)

Questions (a) and (b) refer to the following class:

```
public class A {
    public static void first(int x) {
        try {
            System.out.println("First in");
            second(x);
        } catch (ArithmeticException e) {
            System.out.println("Math!")
        } catch (Exception e) {
            System.out.println("First except");
        }
        System.out.println("First out");
    }

    public static void second(int x) throws Exception {
        System.out.println("Second in");
        if(x < 0) {
            throw new Exception();
        }
        int y = 5/x;
        System.out.println("Second out");
    }
}
```

(a) (3 points)
Write down what the call `first(0)` prints to the console.
First in
Second in
Math!
First out

(b) (3 points)
For what arguments in a call on method `first` will "Second out" be printed? For all $x > 0$, `first(x)` will print "Second out"

(c) (3 points)
Which of the following operations take time proportional to n? Select all that apply.

(i) Look up element $n-1$ in a doubly linked of length n. Assume the list implementation matches the specification given in assignment A3.

(ii) Look up element number i in an array of length n.

(iii) Look up the last element in a singly linked list of length n.

(d) (3 points)
The following code is executed. What gets printed out?

```
char x = 'a'; // (int) 'a' is 97
int y = 103; // (char) 103 is 'g'
String s = x + y + "Number";
System.out.println(s);
```

200Number

(e) (3 points)
Describe in one sentence an advantage of using an array compared to using a doubly linked list.
An array can be used for constant time access to each of its elements

3. Recursion (20 Points)

(a) 5 points Complete function `sumDigitsOddSq` below according to its specification. Do not use class `String`. Do not use a loop. Use only recursion. Solutions using class `String` or a loop will not receive credit.

Solution

```
/** Return the sum of the decimal digits of n, with the odd digits squared.
 *  Example: for n = 36 return 3^2 + 6 = 15
 *  Precondition: n >= 0
 */
public static int sumDigitsOddSq(int n) {
    if (n == 0) {
        return 0;
    }
    int d = n % 10;
    return (d % 2 == 0 ? d : d * d) + sumDigitsOddSq(n / 10);
}
```

(b) 15 points Consider class `Node` below. `Node` is a node of a singly-linked list. The relevant fields are shown below.

```
/** A Node of a Singly-Linked List */
public class Node<T> {
    T value; //The value stored in this Node
    Node<T> next; //The next node in the list, null if this node is last
}
```

Complete function `printMerge` below according to its specification. Do not use the `new` keyword anywhere in your solution. Do not use a loop. Use only recursion. Solutions using the `new` keyword or a loop will not receive credit.

Solution

```
/** Print the values in n1 and n2 in (ascending) sorted order.
  * Precondition: n1 and n2 are both sorted linked lists
  */
public static void printMerge(Node<Integer> n1, Node<Integer> n2) {
    if (n1 == null && n2 == null) {
        return;
    }
    // n1 is empty
    if (n1 == null) {
        System.out.println(n2.value);
        printMerge(n1, n2.next);
        return;
    }
    // n2 is empty or n1 has a lesser value
    if (n2 == null || n1.value < n2.value) {
        System.out.println(n1.value);
        printMerge(n1.next, n2);
        return;
    }
    //Neither are empty and n2 is lesser or equal to n1
    System.out.println(n2.value);
    printMerge(n1, n2.next);
}
```

4. Object Oriented Design (25 points)

Consider the following class that models a city and its tax code. The tax rate is 5% and 1% for commercial and residential structures respectively.
Note: Constructors and some javadoc comments have been omitted for brevity.

```
public class Building {
    public double value;
    public String address;
}

public class Residential extends Building {
    /** Return the amount of taxes owed for this building */
    public double tax() {
        return value * 0.01;
    }
}

public class Commercial extends Building {
    /** Return the amount of taxes owed for this building */
    public double tax() {
        return value * 0.05;
    }
}

public class City {
    private List<Building> buildings;

    /** Return the total property tax revenue from all buildings
     *  in the city */
    public double calculatePropertyTax() {
        double total = 0;
        for (Building b : buildings) {
            if (b instanceof Residential) {
                total = total + ((Residential)b).tax();
            }
            if (b instanceof Commercial) {
                total = total + ((Commercial)b).tax();
            }
        }
        return total;
    }
}
```

(a) 5 points Identify *two* design problems with class `Building` and its subclasses from an object-oriented perspective. Explain why these two are problems in at most 2 sentences per problem. Hint: Think about what we discussed in the recitation on abstract classes. What would happen if we now also wanted to include `Utility` type buildings in addition to `Commercial` and `Residential`?

We accepted any two of the following:

- Access modifiers for fields in building are `public`
- All buildings require a `tax` function, should be an abstract method in building.
- `calculatePropertyTax` cannot support new building types being added

(b) 10 points Modify the above program by improving the object-oriented design elements. At a minimum, your improvements should correct the problems from part (a) and simplify method `calculatePropertyTax`. You may write small changes in-line with the provided code on the previous page instead of copying everything to here. However, if you are rewriting a whole function or class please use the space below. You do not have to write any constructors. *Please write neatly and clearly mark your changes!*

Solution

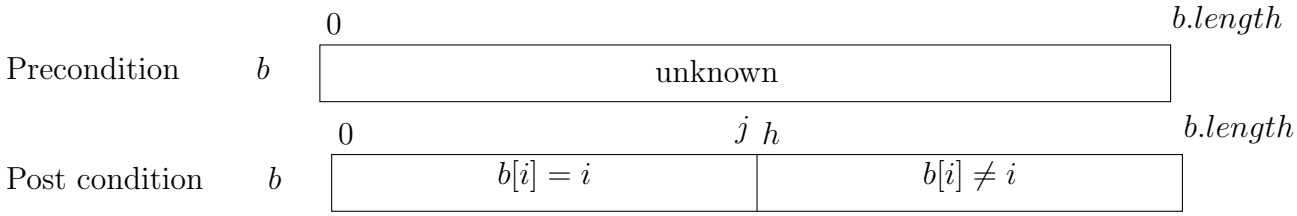
```
public abstract class Building {
    private double value;    // we also accept protected
    private String address;

    /** Return the amount of taxes owed for this building */
    public abstract double tax();

    /** Return the value of this building */
    public double getValue() {return value;}
}

public class Residential extends Building {
    /** Return the amount of taxes owed for this building */
    public double tax() {
        return getValue() * 0.01;    // if protected, using the field
    }                                // directly is fine
}

public class Commercial extends Building {
    /** Return the amount of taxes owed for this building */
    public double tax() {
        return getValue() * 0.05;
    }
}

public class City {
    private List<Building> buildings;

    /** Return the total property tax revenue from all buildings
     *  in the city */
    public double calculatePropertyTax() {
        double total = 0;
        for (Building b : buildings) {
            total = total + b.tax();
        }
        return total;
    }
}
```

(c) 10 points Suppose class `City` has a list of `Buildings`. You want to sort this list so that it is ordered by *decreasing* value. In order to do this, class `Building` must implement the interface `Comparable<Building>`.
Implement function `compareTo` below according to its specification.
Solution

```
public abstract class Building implements Comparable<Building> {
    /**
     * Compare this object with obj. Return a negative integer, zero,
     * or a positive integer depending on whether this object is less
     * than, equal to, or greater than obj.
     */
    public int compareTo(Building obj) {
        double diff = obj.value - value;
        if (diff > 0) {
            return 1;
        }
        if (diff < 0) {
            return -1;
        }
        return 0;
    }
}
```

5. Loop Invariants (20 Points)

Consider the following array: `[0,1,2,3,4,5,6,7,14,15]`. Observe that the first 7 elements in the array have a value equal to their index, and the remaining items have a value that is not equal to their index. You are developing a loop to find the last element with a value equal to its index for an array b that will run in $O(\log b.length)$ time. The precondition and post condition of this loop are provided below:



(a) 8 points Using the precondition and post condition above, develop an invariant you would use to write your loop. Fill in that invariant below. *Note: Please write clearly and legibly. Answers unclear to the grader will receive no credit*

Solution

(b) 12 points Below, write the loop with initialization to truthify the post condition. You must use the invariant you developed above, and your loop must run in $O(\log b.length)$ time. You will receive 3 points for correct initialization, 3 points for a correct while condition, 3 points for proper progress within the loop body, and 3 points for maintaining the invariant.

Solution

```
/** Return the index of the last element that is equal to its index.
 *  Precondition: All elements that are equal to their index come before
 *  all elements that are not
 */
public int findEqualIndex(int[] b) {
    // Initialization
    int j = -1;
    int h = b.length;
    while (h != j + 1) {
        int m = (h + j) / 2;
        if (b[m] == m) {
            j = m;
        } else {
            h = m;
        }
    }
    return j;
}
```