CS/ENGRD2110: Prelim 1

8th of March, 2012

NAME : ______________________________

NETID: ________________

- The exam is **closed book and closed notes**. Do not begin until instructed. You have **90 minutes**. Good luck!
- Start by writing your name and Cornell netid on top! There are **9 numbered pages**. Check now that you have all the pages.
- Web, email, etc. may not be used. Calculator with programming capabilities are not permitted. This exam is **individual work**.
- We have **scrap paper** available, so you if you are the kind of programmer who does a lot of crossing out and rewriting, you might want to write code on scrap paper first and then copy it to the exam, just 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 we provided**. Answers that are not concise might not receive full points.
- 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.

POINTS:

Classes, Interfaces, Types, and Stuff	_______	/ 21
Recursion	_______	/ 16
Trees	_______	/ 16
Asymptotic Complexity	_______	/ 19
Sorting	_______	/ 14
Lists	_______	/ 14
	===========	
Total	_______	/100

1 Classes, Interfaces, and Types

1. Answer the following questions with either true or false. No explanation necessary.

   9 pts.

   - Only classes define valid types in Java.
   - Every class in Java has no more than one superclasses.
   - Every class in Java has exactly one or zero subclasses.
   - Every interface in Java has exactly one or zero supertypes.
   - A cast changes the static type of a variable.
   - The static type of a variable must always be a subtype of the dynamic type.
   - Downcasts can produce runtime errors.
   - Subclasses can access all the methods of its parent class.
   - The dynamic type of an argument to an overloaded method determines which of the methods is selected.

2. Identify the methods that need to be implemented in class `Four` so that the code compiles without error? Do not name methods that do not necessarily have to be implemented.

   4 pts.

```
abstract class One {
        private int a,b,c;
        public abstract int square(Two a);
        public static int triangle() {
           return 1;
        }
}

interface Two{
        public One mOne(int a);
}

interface Three extends Two {
        public Two mTwo(int b);
        public One mOne(float a);
}

abstract class ThreeAndAHalf extends One {
        public float square(Three a) {
                return 1;
        }
}

class Four extends ThreeAndAHalf implements Three {
        ???
}
```

3. Given the interface definitions from above, does the following class definition contain any errors? If yes, what are the errors?

4 pts.

```
class Five extends ThreeAndAHalf {
        public int square() { return a; }
        public static float triangle() { return 2; }
}
```

4. What output does the following program produce?

4 pts.

```
class Song {
        public int id;
        public Song(int a) {
                id = a;
        }
        public int getID() {
                return id;
        }
}

class SmashHit extends Song {
        public SmashHit(int a) {
                super(a);
        }
        public int getID() {
                return id * 10;
        }
}

public class PrelimOne {
        public static void main(String[] args) {
                Song[] songs = new Song[3];
                songs[0] = new Song(1);
                songs[1] = (Song)new SmashHit(2);
                songs[2] = new SmashHit(3);
                for(int i = 0; i < songs.length; i++) {
                        System.out.println( songs[i].getID() );
                        System.out.println( ((Song)songs[i]).getID() );
                }
        }
}
```

2 Recursion

1. What is the output of rec(3,3)?

   8 pts.

```
public static void rec(int a, int b) {
        if (a <= 0 || b <= 0) return;
        if (a <= b) {
                System.out.print (a + " ");
                System.out.print (b + " ");
                rec(a*2, b+3);
        }
        else {
                rec(a, b-4);
                System.out.print (a + " ");
                System.out.print (b + " ");
        }
        return;
}
```

2. Write a method with the signature `public static int sum(int[] array)` that returns the sum of all the elements in the array. Do this without using any loops but use recursion instead. You can create a helper method, but you may not create any static fields in the class.

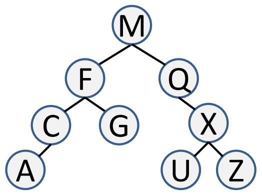
   Hint: Feel free to create a helper methods.

   8 pts.

3 Trees

1. In which order does a preorder, inorder, and postorder traversal print thenodes of the following tree?

   9 pts.

   Preorder:

   Inorder:

   Postorder:

2. Show the binary search tree (BST) after inserting the following sequence of elements. The lexicographical ordering should be used to construct the tree.

   7 pts.

   F X M A Q Z B

4 Asymptotic Complexity

1. Give the mathematical definition of "$f(n)$ is $O(g(n))$".

   5 pts.

2. Answer the following questions with either true or false. No explanation necessary.

   4 pts.

   - MergeSort on an array with n elements has worst-case time complexity $O(n\log(n))$.
   - QuickSort using a random pivot on an array with n elements has worst-case time complexity $O(n\log(n))$.
   - Finding an element in a sorted array with n elements can be done with worst-case time complexity $O(\log(n))$.
   - Inserting an element into an unsorted linked list with n elements can be done in worst-case time $O(1)$.

3. Sort the following functions by their big-O complexity. More formally, order them so that f_i is $O(f_{i+1})$ for all i.
   n^2, 5^n, 1000, $\sqrt{n}$, 2^{2^n}, $\log(n)$, $n^{2/3}$

   4 pts.

4. Prove the following: if $f(n)$ is $O(g(n))$, then $f(n)+g(n)$ is $O(g(n))$.

   6 pts.

5 Sorting

1. Given the sorting method below, what would be printed in the console with the input 3,5,2,6,1,4?

   5 pts.

```
public int superSort(int[] arr){
   int count = 0;
   for (int i = 1; i < arr.length; i++) {
      count++;
      int temp = arr[i];
      int k;
      for (k = i; 0 < k && temp > arr[k-1]; k--){
         count++;
         arr[k] = arr[k-1];
      }
      arr[k] = temp;

      // print out the content of array arr
      String arrString = "";
      for (int j = 0; j < arr.length; j++){
         arrString = arrString + arr[j]  + " ";
      }
      System.out.println(arrString);
   }
   return count;
}
```

2. Give an array of length 6 that would minimize the number returned (i.e., `count`) by superSort. Also, state the value of `count`.

   3 pts.

3. Give an array of length 6 that would maximize the number returned (i.e., `count`) by superSort. Also, state the value of `count`.

   3 pts.

4. Answer with “Yes” or “No” or “Maybe” or “Sometimes”: Is the sorting algorithm from above stable? No explanation necessary.

   3 pts.

6 Lists

1. Assume that the following class `List` implements a doubly-linked list. A doubly-linked list stores not only a pointer to the next `ListNode` (i.e., `next`), but also a pointer to the previous `ListNode` (i.e., `prev`) in the list. Furthermore, it stores a pointer not only to the first element (i.e., `start`), but also to the last element (i.e., `end`) in the header class.
   a) Give one example list for which “mystery()” returns true.
   b) Give one example list for which “mystery()” returns false.
   c) Explain in general what the method “mystery()” does.

   6 pts.

```
class ListNode {
   ListNode prev,next;
   int value;
}

public class List {
    ListNode start,end;

    boolean mystery() {
        return(mystery_helper(start,end));
     }

     boolean mystery_helper(ListNode a, ListNode b) {
        if(a == null) {
           return(true);
        }
        if(a == b) {
           return(false);
        }
        if(a.value == b.value) {
           return(mystery_helper(a.next,b.prev));
        }
        return(false);
     }

   // other methods of doubly-linked list implementation are not shown
}
```

2. Add a method `void insertAfter(int afterThis, int toInsert)` to the class `List` from above. This method should insert `toInsert` in the spot behind the first occurrence of `afterThis` in the list. If the list does not contain `afterThis`, then the list should remain unchanged. Note: Do NOT use other methods in `List`, but write the method from scratch.

8 pts.