CS/ENGRD2110: Prelim 1 SOLUTION

October 2, 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—no calculators should be needed anyway. 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          _______ / 13

Recursion                                      _______ / 18

Trees                                          _______ / 16

Asymptotic Complexity                          _______ / 17

Sorting                                        _______ / 14

Lists                                          _______ / 17

                                               ===========

Total                                          _______ /95
```

1 Classes, Interfaces, and Types

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

   9 pts.

   - Java interfaces do not allow static method definitions.
   - All types in Java are subtypes of Object.
   - All classes in Java implicitly extend the Object class.
   - A cast can change the dynamic type of a variable.
   - A subclass can override a private method of its parent class to make it public.
   - Upcasts can produce runtime errors.
   - The dynamic type of an argument to an overloaded method determines which of the methods is selected.
   - Java interfaces can not contain variable definitions.
   - The dynamic type of a variable must always be a subtype of the static type.

   SOLUTION:
   false, false, true[1], false, false, false, false, false, true
   END SOLUTION

2. What is printed by `System.out.println(new Bar("A"));` for the following two classes?

   4 pts.

```
class Foo {
  String s;

  Foo(String t) {
    s = "C" + t;
  }

  public String toString() {
    return s;
  }
}

class Bar extends Foo {
  Bar(String r) {
    super("B" + r);
  }
}
```

   SOLUTION:
   CBA
   END SOLUTION

[1]We accepted true or false here, since the wording of the question was potentially confusing.

2 Recursion

1. What is the output of `goforit(false,false)`?

10 pts.

```
public static void goforit(boolean a, boolean b) {
   if (!a & !b) {
      goforit(a,!b);
      System.out.println("KABOOM!");
   }
   else {
      if ( a & !b) {
         goforit(a,!b);
      }
      else if (!a &  b) {
         goforit(!a, !b);
      }
      System.out.println(a + " " + b);
   }
}
```

```
SOLUTION:
true true
true false
false true
KABOOM!
END SOLUTION
```

2. Write a method with the signature `public static int sumOfSquares(int[] x)` that returns the sum of all the array elements' squared values. 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.

8 pts.

```
SOLUTION:
       public static int sumOfSquares(int[] x) {
               return sumOfSquares(x,x.length-1);
       }

       public static int sumOfSquares(int[] x, int end) {
               if(end == -1) {
                   return(0);
               }
               else {
                   return x[end]*x[end] + sumOfSquares(x, end-1);
               }
       }
END SOLUTION
```

3 Trees

1. Given the sequence of integers (4, 1, 10, 5, 2, 14), consider a binary search tree (BST) constructed by inserting the sequence of integers at the next available location (using the usual increasing numerical ordering, with lower values on the left, and larger values on the right). Draw a diagram of the binary search tree that you would obtain.

7 pts.

```
   4
  / \
 /   \
1     10
 \    /\
  2  5  14
```

2. What sequence of elements would you obtain if you traversed this binary search tree (BST) using (i) Preorder traversal, (ii) Inorder traversal, and (iii) Postorder traversal?

9 pts.

Preorder: 4 1 2 10 5 14

Inorder: 1 2 4 5 10 14

Postorder: 2 1 5 14 10 4

4 Asymptotic Complexity

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

8 pts.

- $n^2 = O(n^n)$
- $\sqrt{n} = O(\log(n))$
- $\sqrt{n} = O(2^{2^n})$
- $n^{\log(n)} = O(2^{2^{1000}})$
- Inserting an element into a sorted linked list with n elements can be done in worst-case time $O(\log(n))$.
- The worst-case time complexity of MergeSort is the same as QuickSort.
- SelectionSort has expected-case time complexity of $O(n \log(n))$.
- InsertionSort has worst-case time complexity of $O(n \log(n))$.

SOLUTION:
true, false, true, false, false, false, false, false
END SOLUTION

2. Using the definition of big O notation, prove that $f(n) = \frac{1}{n} + n\log(n) + n^2$ is $O(n^2)$ by finding a suitable witness pair (c, N).

9 pts.

SOLUTION:
By definition, we need to find constants c and N such that for all $n \geq N$ it holds that

$$f(n) = \frac{1}{n} + n\log(n) + n^2 \leq cn^2.$$

Consider each term separately. First, consider that

$$\frac{1}{n} \leq 1 \leq n^2, \quad \text{for} \quad n \geq 1.$$

Second, consider that

$$n\log(n) \leq n^2, \quad \text{for} \quad n \geq 1.$$

It follows that for $n \geq N = 1$ we have

$$\frac{1}{n} + n\log(n) + n^2 \leq n^2 + n^2 + n^2,$$

and so that $f(n) \leq 3n^2$ for $n \geq 1$. Hence we can use $(c, N) = (3, 1)$ as a witness pair.
END SOLUTION

5 Sorting

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

5 pts.

```
public int crimeSort(int[] lineup){

   int n = lineup.length;
   int i,j;
   int suspect;
   int crimes = 0;

   for (j = 0; j < n-1; j++) {

      suspect = j;
      for ( i = j+1; i < n; i++) {
         if (lineup[i] < lineup[suspect]) {
            suspect = i;
         }
      }
      if ( suspect != j ) {
         int accomplice  = lineup[j];
         lineup[j]       = lineup[suspect];
         lineup[suspect] = accomplice;
         crimes++;
      }

      // Print out "lineup":
      String s = "";
      for (int j = 0; j < n; j++){
         s = s + lineup[j]  + " ";
      }
      System.out.println(s);
   }

   return crimes;
}
```

SOLUTION:
1 5 2 6 3 4
1 2 5 6 3 4
1 2 3 6 5 4
1 2 3 4 5 6
1 2 3 4 5 6 (Not required: note that the method is SelectionSort)
END SOLUTION

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

3 pts.

SOLUTION:
Any array in ascending order will give a count of 0, e.g., [1,2,3,4,5,6].

END SOLUTION

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

3 pts.

SOLUTION:
You could report any length-6 array which causes SelectionSort to do find a swap (a "crime") each pass. Note that a reverse-sorted array does not give the max value. Although you only need to state one case, here are the 120 arrays that give the maximum value of 5:
END SOLUTION

```
2 3 4 5 6 1
2 3 4 6 1 5
2 3 5 1 6 4
2 3 5 6 4 1
2 3 6 1 4 5
2 3 6 5 1 4
2 4 1 5 6 3
2 4 1 6 3 5
2 4 5 3 6 1
2 4 5 6 1 3
2 4 6 3 1 5
2 4 6 5 3 1
2 5 1 3 6 4
2 5 1 6 4 3
2 5 4 1 6 3
2 5 4 6 3 1
2 5 6 1 3 4
2 5 6 3 4 1
2 6 1 3 4 5
2 6 1 5 3 4
2 6 4 1 3 5
2 6 4 5 1 3
2 6 5 1 4 3
2 6 5 3 1 4
3 1 4 5 6 2
3 1 4 6 2 5
3 1 5 2 6 4
3 1 5 6 4 2
3 1 6 2 4 5
3 1 6 5 2 4
3 4 2 5 6 1
3 4 2 6 1 5
3 4 5 1 6 2
3 4 5 6 2 1
3 4 6 1 2 5
3 4 6 5 1 2
3 5 2 1 6 4
3 5 2 6 4 1
3 5 4 2 6 1
3 5 4 6 1 2
3 5 6 1 4 2
3 5 6 2 1 4
3 6 2 1 4 5
```

```
3  6  2  5  1  4
3  6  4  2  1  5
3  6  4  5  2  1
3  6  5  1  2  4
3  6  5  2  4  1
4  1  2  5  6  3
4  1  2  6  3  5
4  1  5  3  6  2
4  1  5  6  2  3
4  1  6  3  2  5
4  1  6  5  3  2
4  3  1  5  6  2
4  3  1  6  2  5
4  3  5  2  6  1
4  3  5  6  1  2
4  3  6  2  1  5
4  3  6  5  2  1
4  5  1  2  6  3
4  5  1  6  3  2
4  5  2  3  6  1
4  5  2  6  1  3
4  5  6  2  3  1
4  5  6  3  1  2
4  6  1  2  3  5
4  6  1  5  2  3
4  6  2  3  1  5
4  6  2  5  3  1
4  6  5  2  1  3
4  6  5  3  2  1
5  1  2  3  6  4
5  1  2  6  4  3
5  1  4  2  6  3
5  1  4  6  3  2
5  1  6  2  3  4
5  1  6  3  4  2
5  3  1  2  6  4
5  3  1  6  4  2
5  3  4  1  6  2
5  3  4  6  2  1
5  3  6  1  2  4
5  3  6  2  4  1
5  4  1  3  6  2
5  4  1  6  2  3
5  4  2  1  6  3
5  4  2  6  3  1
5  4  6  1  3  2
5  4  6  3  2  1
5  6  1  2  4  3
5  6  1  3  2  4
5  6  2  1  3  4
5  6  2  3  4  1
5  6  4  1  2  3
5  6  4  2  3  1
6  1  2  3  4  5
6  1  2  5  3  4
6  1  4  2  3  5
```

```
6 1 4 5 2 3
6 1 5 2 4 3
6 1 5 3 2 4
6 3 1 2 4 5
6 3 1 5 2 4
6 3 4 1 2 5
6 3 4 5 1 2
6 3 5 1 4 2
6 3 5 2 1 4
6 4 1 3 2 5
6 4 1 5 3 2
6 4 2 1 3 5
6 4 2 5 1 3
6 4 5 1 2 3
6 4 5 3 1 2
6 5 1 2 3 4
6 5 1 3 4 2
6 5 2 1 4 3
6 5 2 3 1 4
6 5 4 1 3 2
6 5 4 2 1 3
```

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

3 pts.

SOLUTION:
Yes. (The algorithm (Selection Sort) only swaps two elements if one is strictly greater than the other. That means that two equal elements are never swapped and therefore never change their order.)
END SOLUTION

6 Lists

1. Assume that the following class `List` implements a singly-linked list comprised of `ListNode` objects. Write a `List` method "`int getSumOfSquares()`" that computes the sum of squares of the list's values.

8 pts.

```
class ListNode {
   public ListNode next;
   public int      value;

   public ListNode() {}
}

public class List {
    public ListNode start;

    public List() {}

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

SOLUTION:
int getSumOfSquares()
{
   ListNode x   = start;
   int      sum = 0;
   while(x != null) {
      sum += x.value * x.value;
      x = x.next;
   }
   return sum;
}
END SOLUTION:
```

2. Add a method `void removeOddEntries()` to the `List` class from above. This method should remove any `ListNode` objects corresponding to odd-valued entries. If the list does not contain any odd entires, then the list should remain unchanged. Note: Do NOT use other methods in `List`, but write the method from scratch.

9 pts.

```
public void removeOddEntries()
{
   // REMOVE ODD NODES AT THE BEGINNING:
   while(start != null && isOdd(start.value)) {
      start = start.next;
   }
   if(start==null) return;

   // NEXT REMOVE ODD NODES IN THE MIDDLE:
   // DURING LOOP x is nonnull and even; xNext may be odd (or null).
   ListNode x = start; //assert: x.value is even and nonnull
   while(x.next != null) {//step through list
      if(isOdd(x.next.value)) {//remove x.next
         x.next = x.next.next; // may be null or odd
      }
      else {// both x and x.next are nonnull & even --> move forward
         x = x.next;
      }
   }
}

private static boolean isOdd(int value) {
   return (value%2 != 0);//remainder when divided by 2 is nonzero
}
```