Name: _____

CSSE 220—OBJECT-ORIENTED SOFTWARE DEVELOPMENT

FINAL EXAM, WINTER, 2007-08 WRITTEN QUESTIONS

The "written" part of this exam is entirely on paper. You are not allowed to use any resources other than the one sheet of paper you were told you could bring,

I recommend spending no more than 90 minutes on this part (most people should finish it much sooner).

THE PROBLEMS

1. (12 points) Consider the following initial array configuration:

11	5	3	8	7	1	10	2	4	12	6	9

(a) If the bubble sort algorithm from class is applied to this array, show the state of the array immediately following the first **two** executions of the outer loop.

(b) If the insertion sort algorithm from class is applied to the initial array, show the state of the array immediately following the first **four** executions of the outer loop.

(c) If the selection sort algorithm from class (selecting the largest elements first) is applied to the initial array, show the state of the array immediately following the first **two** executions of the outer loop.

(d) If the merge sort algorithm from class is applied to the initial array, show the state of the array immediately before the final call to `merge()`.

(e) If the Shell sort algorithm is applied to the initial array, and the first gap size is 4, show the state of the array immediately after that partial sort with the first gap size.

(f) How many inversions are in the initial array? _____

2. (5 points) When we wrote insertion sort in class, we started at the beginning of the array, and the outer loop went in forward order. We could write an insertion sort starts at the end of the array, having the outer loop go in reverse order. Write it.

```
public void sort (int [] a) {

}
```

3. (8 points) Mark each of the following statements as True (T) or False (F) by circling the letter. For those that are False, briefly explain why or give a counterexample.

T F If f(N) is O(g(N)), then g(N) is O(f(N)).

T F If f(N) is Θ(g(N)), then g(N) is Θ(f(N)).

T F 2^N is O(3^N).

T F 3^N is O(2^N).

T F $\log_3 N$ is $\Theta(\log_2 N)$

T F If the limit of f(N)/g(N) as N$\rightarrow\infty$ is 5, then f(N) is O(g(N))

4. (10 points) The background for this problem is on this page; the actual problem is on the following page.

Consider the following method description from **the java.util.List** documentation:

`List<E>` **`subList`**`(int fromIndex, int toIndex)`

Returns a view of the portion of this list between the specified `fromIndex`, inclusive, and `toIndex`, exclusive. (If `fromIndex` and `toIndex` are equal, the returned list is empty.) The returned list is backed by this list, so non-structural changes in the returned list are reflected in this list, and vice-versa. The returned list supports all of the optional list operations supported by this list.

This method eliminates the need for explicit range operations (of the sort that commonly exist for arrays). Any operation that expects a list can be used as a range operation by passing a subList view instead of a whole list. For example, the following idiom removes a range of elements from a list:

```
list.subList(from, to).clear();
```

Similar idioms may be constructed for `indexOf` and `lastIndexOf`, and all of the algorithms in the `Collections` class can be applied to a subList.

The semantics of the list returned by this method become undefined if the backing list (i.e., this list) is *structurally modified* in any way other than via the returned list. (Structural modifications are those that change the size of this list, or otherwise perturb it in such a fashion that iterations in progress may yield incorrect results.)

Parameters: `fromIndex` - low endpoint (inclusive) of the subList
`toIndex` - high endpoint (exclusive) of the subList

Returns: a view of the specified range within this list

Throws: `IndexOutOfBoundsException` - for an illegal endpoint index value
`(fromIndex < 0 || toIndex > size || fromIndex > toIndex)`

Here is an example of the use (and misuse) of the `subList` method:

```
public static void main(String[] args) {
   ArrayList<Integer> list = new ArrayList<Integer>();
   for (int i=0; i< 12; i++) {
      list.add(2*i+1);
   }

   List<Integer> subList = list.subList(3, 7);
   System.out.println(subList);
   list.set(4, 100);
   System.out.println(subList);
   list.add(4, -6);
   System.out.println(subList);

}
```

Output:

```
[7, 9, 11, 13]
[7, 100, 11, 13]
Exception in thread "main" java.util.ConcurrentModificationException
      at java.util.SubList.checkForComodification(Unknown Source)
      at java.util.SubList.listIterator(Unknown Source)
      at java.util.AbstractList.listIterator(Unknown Source)
      at java.util.SubList.iterator(Unknown Source)
      at java.util.AbstractCollection.toString(Unknown Source)
      at java.lang.String.valueOf(Unknown Source)
      at java.io.PrintStream.println(Unknown Source)
    at SublistExperiment.main(SublistExperiment.java:19)
```

4. (continued)

(a) (5 points) Describe in a few sentences and/or code how you would modify our representation of ArrayList in order to support and implement this **subList** method and the rest of the described behavior.

(b) (5 points) Describe in a few sentences and/or code how you would modify our representation of LinkedList in order to support and implement this **subList** method and the rest of the described behavior.

Name: _____

5. (10 points) In a homework problem, we considered how to efficiently store and access the elements of an NxN lower triangular matrix. Suppose we want to represent N x N upper-triangular matrices instead. Since we know that M[i][j] is always zero if j<i, we only need to store the elements for which j≥i. We can store those elements in a one-dimensional array, **elements**. Something like this:

```
public class UpperTriangular {

   private int[] elements;
   private int dimension;

   public UpperTriangular(int n) {
      elements = new int[  ??? ];
      dimension = n;
   }
```

(a) (3 points) What expression should go inside the brackets (where the question-marks are)?

On Exam 2, we saw that the formula for looking up matrix elements in the array is simplest if we store the nonzero elements column-by-column. Suppose that some other constraint forces us to store them row-by-row.

(b) (2 points) In this representation of a 5x5 upper-triangular matrix, at what index in the **elements** array would M[2][4] be stored? _____

It is still possible to write a constant-time formula for finding M[i][j].

```
   public int get(int row, int col){
      if (col < row)
         return 0;
      return elements[   ???     ];
   }
```

(c) (5 points) What expression should go inside the brackets (where the question-marks are)?

6. (3 points) On what basis would you decide between using a HashMap and using a TreeMap to store a table of keys and their associated values?

Name: _____

7. (6 points) The following is an excerpt from the **BankServer.java** code that we discussed in class.

```
ServerSocket server = new ServerSocket(8888);

 while (true) {
   Socket s = server.accept();
   BankService service = new BankService(s, bank);
   Thread t = new Thread(service);
   t.start();
}
```

(a) What does the number 8888 specify here?

(b) What is a **ServerSocket** used for?

(c) What does a call to the **accept** method do?

(d) From what you see in this code, what interface must the **BankService** class implement?

(e) What will a **service** object have to do with the socket that is passed to its constructor before it can begin communicating with a client.

(f) Why does this program create and start threads inside an infinite loop?

8. (a) (5 points) Write a Comparator (i.e. **java.util.Comparator**, which we discussed frequently) called `CompareStringsIgnoreSpaces` that compares two strings of characters, while ignoring any spaces in the strings. For example, in this ordering, "abc" and "ab c" are equal, while "ab c" comes after "abb" (the former would come before the latter in the normal **String** ordering).

(b) (3 points) **java.util.Arrays.sort()** takes an array of objects and a Comparator that can compare those objects. Show how you would call it using an array of Strings, named **strings**, and a **CompareStringsIgnoreSpaces** Comparator, in order to sort the array in the order implied by the description in part (a).

9. (2 points) What is Polymorphism? Given an example of something we did where Polymorphism made the code more general or easier to write.

Name: _____

10. (6 points) Write a simple method that prints all of the elements from an arbitrary collection in reverse order, by taking advantage of the properties of one of the data structures that we studied.

```
public static void printReverse (Collection c) {




}
```