CMP 338 (Fall 2011)

Exam 1, 10/6/11

Name (sign) ______________________________

Name (print) ______________________________

email ______________________________

Question	**Score**
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
TOTAL	

1 a) The signature of a Java method to sort an array of **`doubles`** into ascending order is provided below. Complete the method to implement **Insertion Sort**. Do ***not*** call any library methods. You *may* use your own helper methods, provided you also write them.

```
static void insertionSort(double[] a)
```

1 b) In the worst case, how many comparisons does insertion sort require to sort N items? (Use tilda notation. If you know the leading constant, provide it. Otherwise, use "c".)

1 c) Briefly characterize the worst case input for insertion sort.

1 d) Under what circumstances does insertion sort provide acceptable performance?

2 a) The signature of a Java method to sort an array of **`doubles`** into ascending order is provided below. Complete the method to implement **Merge Sort**. Do ***not*** call any library methods. You *may* use your own helper methods, provided you also write them. You may also call the method **`merge`** assuming that it will do what it is supposed to do.

```
static void mergeSort(double[] a)
```

2 b) In the worst case, how many comparisons does merge sort require to sort N items? (Use tilda notation. If you know the leading constant, provide it. Otherwise, use "c".)

2 c) How much extra space does mergesort require?

2 d) Write the signature for the **`merge`** method.

2 e) Briefly, what is the **`merge`** method supposed to do?

3 a) The signature of a Java method to sort an array of **`doubles`** into ascending order is provided below. Complete the method to implement **Quicksort**. You may use methods of the **`Random`** library, but no other library methods. You *may* use your own helper methods, provided you also write them. You may also assume that the **`partition`** method does what it is supposed to do.

```
static void quicksort(double[] a)
```

3 b) In the worst case, how many comparisons does quicksort require to sort N items? (Use tilda notation. If you know the leading constant, provide it. Otherwise, use "c".)

3 c) Give three reasons why you might ***not*** want to use quicksort to sort a list of Items.

4 a) The signature of a Java method to sort an array of **doubles** into ascending order is provided below. Complete the method to implement **Heap Sort**. You *may* call the methods (and constructor) of a **MinPQ** priority queue library. You *may* use your own helper methods, provided you also write them.

```
static void heapSort(double[] a)
```

4 b) In the worst case, how many comparisons does heap sort require to sort N items? (Use tilda notation. If you know the leading constant, provide it. Otherwise, use “c”.)

4 c) How much extra space is required by (the array version of) heap sort?

4 d) In the worst case, how many comparisons are required to remove the smallest item from a **MinPQ** (and restore its heap) containing N items?

5 a) The signature of a Java method to return the **`k`**-th smallest entry in an array of N **`doubles`** is provided below. Complete the method to return its result using (on average) ~ c N comparisons. Do ***not*** call any library methods. You *may* use your own helper methods, provided you also write them. You may also assume that the **`partition`** method does what it is supposed to do.

```
static void select(double[] a, int k)
```

5 b) In the worst case, how many comparisons does your **`select`** method require to sort N items? (Use tilda notation. If you know the leading constant, provide it. Otherwise, use "c".)

5 c) Describe a worst case input for your **select** method.

5 d) In the worst case, how many comparisons are required by the best possible implementation of the **`select`** method?

6) The signature of a Java method is provided below. It takes two arrays and an `int` as parameters. The first array is sorted. Implement the method to look up the third parameter in the first array and return the corresponding value from the second array. If the arrays have N elements, your method may use ~ c lg N comparisons. Do ***not*** call any library methods. You *may* use your own helper methods, provided you also write them.

```
double getWeight(int[] sn, double[] weight, int n)
```

6 a) How much extra space is required by shell sort?

6 b) Under what circumstances would one ***not*** want to use shell sort?

7) Under what circumstances is bubblesort acceptable?

8 a) What does it mean for a sort to be ***stable***?

8 b) Which of the sorts we have studied are stable?

8 c) Under what circumstances would one want to use a stable sort?

9 a) What does Java's **`Comparable`** interface enable?

9 b) What does Java's **`Comparator`** interface enable?

10 a) Complete the following method to determine the number of distinct integers in an array. (You may call library methods.)

```
static int unique(int[] a)
```

10 b) What is your method's expected asymptotic running time?