University of Maryland, College Park
Dept. of Computer Science
CMSC132
Midterm #3

First Name (PRINT): _____

Last Name (PRINT): _____

University ID: _____

Section/TAName: _____

I pledge on my honor that I have not given or received any unauthorized assistance on this examination.

Your signature: _____

Instructions

- *This is a closed-book and closed-notes exam.*
- ***Total point value is 70 points.***
- *Thus is a 50 minute exam.*
- ***WRITE NEATLY**. If we cannot understand your answer, we will not grade it (i.e., 0 credit).*

Grader Use Only

Page 2	(14)	
Page 3	(12)	
Page 4	(12)	
Page 5	(14)	
Page 6	(6)	
Page 7	(12)	
Total	**(70)**	

1. (2 points) Which of the following sorting algorithms are "comparison-based"? (Circle all that apply.)

   Heapsort **Bubble Sort** **Bucket Sort** **Radix Sort**

2. (2 points) Which of the following algorithms are "in place"? (Circle all that apply).

   Tree Sort **Selection Sort** **Bucket Sort** **Counting Sort**

3. (2 points) What is the fastest (average case) running time possible for a comparison-based sorting algorithm? (Use big-O notation.)

4. (2 points) Which sorting algorithm is typically used by Bucket Sort, to sort each bucket after the elements have been distributed?

5. (6 points) Implement Bubble Sort in the space below. (When the method is over, the array referenced by the parameter, **a**, should be sorted.)

```
public static void bubbleSort(Comparable[] a) {
```

6. (6 points) Fill in the table below with big-O running times for the average case and the worst case of each sorting algorithm, using the variable **n** to represent the size of the data set. (Use the smallest category you can, and use the usual notation.)

	Average Case	**Worst Case**
Counting Sort (Assume that the *range* of the data is no larger than n.)	O( )	O( )
Selection Sort	O( )	O( )
Bucket Sort (Assume that the *range* of the data is at least as large as n, and that the data is generated randomly, but uniformly over the range.)	O( )	O( )
Bubble Sort	O( )	O( )
Quicksort	O( )	O( )
Tree Sort (Using a standard binary search tree.)	O( )	O( )
Merge Sort	O( )	O( )
Heap Sort	O( )	O( )

7. (6 points) Use Radix Sort to alphabetize the Strings below. (Show the order of the elements after each pass through the loop.)

BCA

CCB

ABA

CAB

BBC

8. (2 points) Which sorting algorithm is typically used by Radix Sort for each pass through the loop?

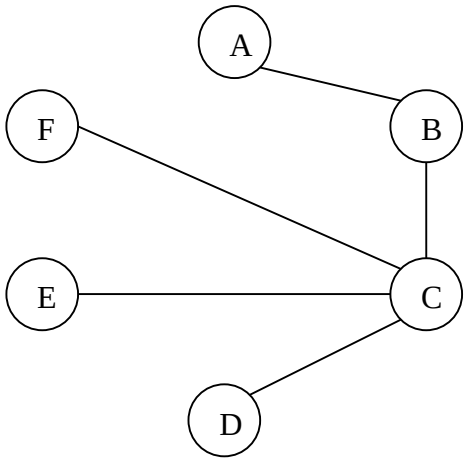
9. (2 points) Which sorting algorithm requires you to select a pivot before moving the data?

10. (2 points) Which sorting algorithm requires you to divide the data into two equal halves before making recursive calls?

11. (6 points) Implement Heap Sort in the space below. (When the method is over, the array referenced by the parameter, `a`, should be sorted.) Assume that the API for the `Heap` class is as follows:

    - `public void add(Comparable element)`
    - `public Comparable removeSmallest()`

```
public static void heapSort(Comparable[] a, Heap h) {
```



12. (3 points) List the vertices in the order they would be visited during a Depth-First Traversal, starting at vertex E. (More than one answer is possible.)

13. (3 points) List the vertices in the order they would be visited during a Breadth-First Traversal, starting at vertex E. (More than one answer is possible.)

14. (2 points) Below is a declaration of a data structure that could be used to implement a *weighted* graph:

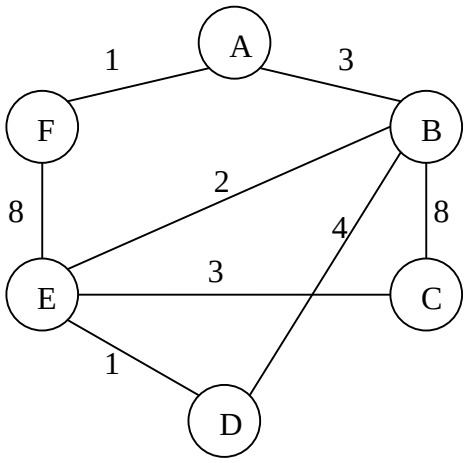
```
HashMap<V, HashMap<V, Integer>> graph;
```

In the space below, declare a similar data structure that could be used to implement an *un-weighted* graph:

15. (4 points) When using an adjacency matrix to implement a graph:

- What type of elements would be in the array if the graph is *weighted*?
- What type of elements would be in the array if the graph is *un-weighted*?

16. (2 points) **HARDEST QUESTION ON THE EXAM:** Calculate the maximum number of edges possible in a directed graph with `n` vertices. (Assume that there is never an edge leading from a vertex to itself.)



17. (6 points) Apply Dijkstra's algorithm using **B** as the starting (source) vertex. Using the tables below, indicate the best known cost/distance and predecessor for each vertex after two successive passes through the algorithm's main (outer) loop. Don't leave any of the boxes blank or you will lose points!

Starting table (filled in for you):

Vertex	A	B	C	D	E	F
Cost	∞	0	∞	∞	∞	∞
Predecessor	-	-	-	-	-	-

Fill in the table below with the best known costs and predecessors after the first pass through the main (outer) loop.

Vertex	A	B	C	D	E	F
Cost						
Predecessor						

Fill in the table below with the best known costs and predecessors after the second pass through the main (outer) loop.

Vertex	A	B	C	D	E	F
Cost						
Predecessor						

18. (2 point) When programming with multiple threads, what do we call the troublesome situation where more than one thread tries to modify a shared data value?

19. (4 points) There are two ways to create threads in Java:

    a. One way is to write a class the extends the class _____.

    b. The other way is to write a class that implements the interface _____.

20. (6 points) Fill in the class below. The `main` method should spawn 500 threads. Each thread will simply output the word "Hi" on the console. (Hint: Use a nested class to define the code that will run in your threads.)

```
public class SillyThreadExample {

   public static void main(String[] args) {

   }
}
```