Name: ______________________________________

Email address (UWNetID): ______________________________________

<u>CSE 332 Winter 2019 Final Exam</u>

(closed book, closed notes, no calculators)

Instructions: Read the directions for each question carefully before answering. We may give partial credit based on the work you **write down**, so show your work! Use only the data structures and algorithms we have discussed in class so far. Writing after time has been called will result in a loss of points on your exam.

Note: For questions where you are drawing pictures, please circle your final answer.

You have 1 hour and 50 minutes, work quickly and good luck!

Total: Time: 1 hr and 50 minutes.

Question	**Max Points**	**Topic**
1	10	Hashing
2	10	Graphs
3	8	More Graphs
4	10	|| Prefix
5	14	ForkJoin
6	12	Concurrency
7	16	Sorting
8	10	P/NP
Total	90	

1) [10 points total] Hash Tables

For a) and b) below, insert the following elements in this order: 7, 9, 48, 8, 37, 57. For each table, TableSize = 10, and you should use the primary hash function h(k) = k%10. If an item cannot be inserted into the table, please indicate this and continue inserting the remaining values.

a) [1 pt] Quadratic probing hash table

0	
1	
2	
3	
4	
5	
6	
7	
8	
9	

b) **[1 pt] Separate chaining hash table** – Use a linked list for each bucket. Order elements within buckets in any way you wish.

0	
1	
2	
3	
4	
5	
6	
7	
8	
9	

c) **[2 pts]** In a sentence or two, describe **double hashing**.

d) **[4 pts]** List 2 cons of **quadratic probing** and describe how one of those is fixed by using **double hashing**.

e) **[2 pts]** Give a tight big-O bound for the worst case runtime of an *Insert in a **separate chaining hash table** containing N elements where each bucket points to its own binary search tree?* **<u>For any credit explain your answer briefly.</u>**

2) [10 points total] Graphs!

a) **[1 pts]** Draw a directed graph containing 4 vertices that has exactly one topological sort.

b) **[3 pts]** You are given Dijkstra's algorithm implemented using a priority queue as described in lecture. The priority queue implementation you must use has the following worst case running times: insert: O(N), deletemin: O(N), decreasekey: O(N^2), increasekey: O(N^2), buildheap: O(NlogN). Given that you must use this priority queue, what is the worst case running time of Dijkstra's? Give your answer as a tight Big-O bound in terms of V and E. **<u>Explain how you got your answer briefly.</u>**

c) **[2 pts]** What is the minimum number of edges in a complete directed graph with 4 vertices? Do NOT include self-loops in your count. **<u>Draw a picture of your graph.</u>**

Minimum Number of Edges:

d) **[2 pt]** We covered two data structures to represent graphs. Give the name of the one with a faster worst case running time for the operation of <u>deleting a particular edge</u> and its running time.

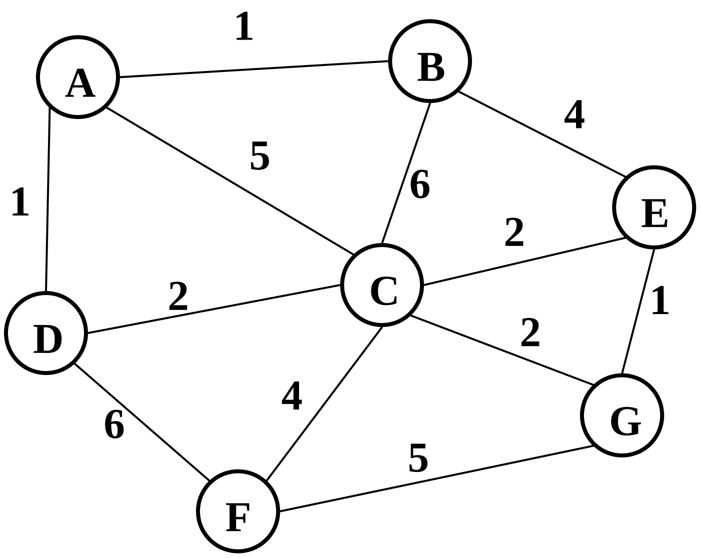
Name of structure: ______________________________________

Worst Case tight Big-O bound on Running Time: __________________________

e) **[1 pt]** Give one way in which Breadth First Search is better than Depth First Search.

f) **[1 pt]** Give one way in which Depth First Search is better than Breadth First Search.

3) [8 points total] More Graphs!
a) [6 pts] Find a minimum spanning tree with Prim's algorithm **using vertex A as the starting node. ****Mark, circle, or highlight edges in the graph that are in your minimum spanning tree. ** Show your steps in the table by crossing through values that are replaced by a new value. Break ties by choosing the letter that comes first alphabetically; ex. if Y and Z were tied, you should pick Y. *Note that b) asks you to recall what order vertices were declared known*.



Vertex	Known (F/T)	Cost	Prev
A			
B			
C			
D			
E			
F			
G			

b) [1 pt] In what order would Prim's algorithm mark each node as *known*?

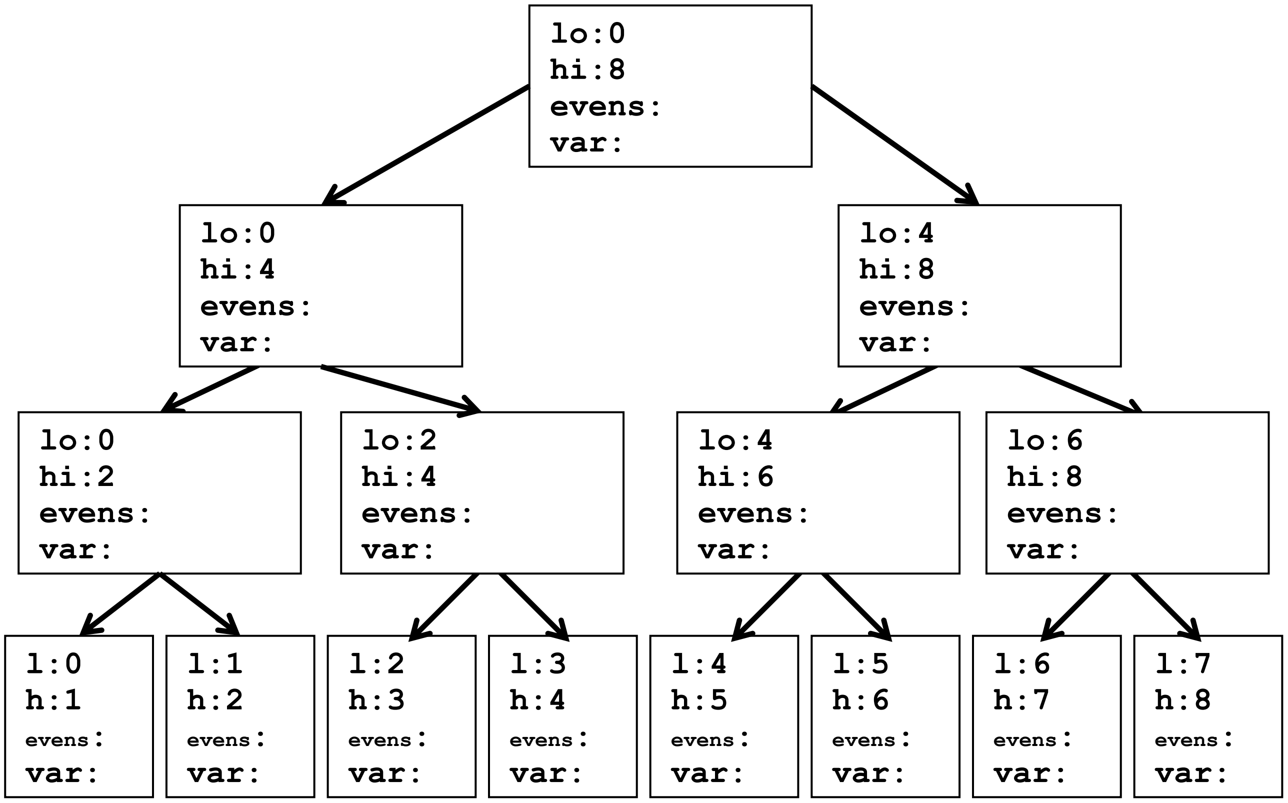
c) **[1 pt]** Will Prim's starting at vertex A find a correct minimum spanning tree if the weight of edge (E, G) is set to be -8? (circle one)

YES **NO**

Did you highlight your original MST in the graph above?

4) [10 points] Parallel "Suffix Count Evens" (Like Prefix, but <u>from the Right</u> instead):

a) Given the following array as input, calculate the "suffix count of evens" using an algorithm similar to the parallel prefix algorithm discussed in lecture. In other words, **`output[i]`** should contain the count of even numbers from **`input[i]`** to **`input[input.length - 1]`** inclusive. The first pass of the algorithm is similar to the first pass of the parallel prefix code you have seen before. Fill in the values for **`evens`** and **`var`** in the tree below. The **`output`** array has been filled in for you. Do not use a sequential cutoff.



Index	0	1	2	3	4	5	6	7
`input`	**0**	**-6**	**7**	**25**	**14**	**14**	**4**	**5**
`output`	**5**	**4**	**3**	**3**	**3**	**2**	**1**	**0**

b) Give formulas for the following values where **`p`** is a reference to a non-leaf tree node and **`leaves[i]`** refers to the leaf node in the tree visible just above the corresponding location in the **`input`** and **`output`** arrays in the picture above.

`p.evens` =

`p.left.var` =

`p.right.var` =

`output[i] =`

c) Describe how you assigned a value to **`leaves[i].evens`** .

5) [14 points] In Java using the ForkJoin Framework, write code to solve the following problem:

- **Input**: An array of `ints` containing values in the range 0-15 (inclusive). There can be duplicates and assume that the length of the array is an odd number.
- **Output**: Return the median value found in the array.

If the input array is {0}, the program would return 0.
If the input array is {15, 3, 6}, the program would return 6.
If the input array is {13, 12, 0, 5, 3, 12, 15, 2, 7}, the program would return 7.

- Do **not** employ a sequential cut-off: **the base case should process one element**. (You can assume the input array will contain at least one `int`.)
- Give a class definition, `FindMedianTask`, **along with any other code or classes needed**.
- Fill in the function `findMedian` **below.**

***You may NOT use any global data structures or synchronization primitives (locks).**
***Make sure your code has O(log n) span and O(n) work.**

```
import java.util.concurrent.ForkJoinPool;
import java.util.concurrent.RecursiveTask;
import java.util.concurrent.RecursiveAction;

class Main{
     public static final ForkJoinPool fjPool = new ForkJoinPool();

     // Returns the median value in input array with values 0-15.
     // Assumes an odd number of values in input array.
     public static int findMedian (int[] input) {

         // Your code here:









     }
}
```

Please fill in the function above and write your class(es) on the next page.

5) (Continued) Write your class(es) on this page.

6) [12 points total] Concurrency: The `PhoneMonitor` class tries to help manage how much you use your cell phone each day. Multiple threads can access the same `PhoneMonitor` object. Remember that synchronized gives you reentrancy.

```
 1
 2 public class PhoneMonitor {
 3    private int numMinutes = 0;
 4    private int numAccesses = 0;
 5    private int maxMinutes = 200;
 6    private int maxAccesses = 10;
 7    private boolean phoneOn = true;
 8    private Object accessesLock = new Object();
 9    private Object minutesLock = new Object();
10
11    public void accessPhone(int minutes) {
12
13       if (phoneOn) {
14
15          synchronized (accessesLock) {
16
17             synchronized (minutesLock) {
18
19                numAccesses++;
20                numMinutes += minutes;
21                checkLimits();
22             }
23          }
24       }
25    }
26
27    private void checkLimits() {
28
29       synchronized (minutesLock) {
30
31          synchronized (accessesLock) {
32
33             if ( (numAccesses >= maxAccesses) ||
34                  (numMinutes >= maxMinutes) ) {
35                     phoneOn = false;
36             }
37          }
38       }
39    }
40 }
```

a) **[4 pts]** Does the `PhoneMonitor` class as shown above have (circle **<u>all</u>** that apply):

a race condition, potential for deadlock, a data race, none of these

Justify your answer. Refer to line numbers in your explanation. <u>Be specific!</u>

6) (**Continued**)

b) **[4 pts]** Suppose we made the `checkLimits` method **`public`**, and changed nothing else in the code. Does this modified `PhoneMonitor` class have (circle **all** that apply):

a race condition, potential for deadlock, a data race, none of these

If there are any FIXED problems, describe why they are FIXED. If there are any NEW problems, give an example of when those problems could occur. Refer to line numbers in your explanation. Be specific!

c) **[4 pts]** Assuming the `checkLimits` method is now **`public`**, add these two methods to the class:

```
public int increaseMaxMinutes(int minutes) {


   maxMinutes += minutes;

   return maxMinutes;

}

public int increaseMaxAccesses(int accesses) {


   maxAccesses += accesses;

   return maxAccesses;

}
```

Now modify the **code above and on the previous page** to ***allow the most concurrent access*** and to avoid all of the potential concurrency problems listed above. Use only `synchronized` statements or methods. Create other objects or fields as needed. **For full credit you must allow the most concurrent access possible without introducing any of the synchronization problems listed above.**

7) [16 points total] Sorting

a) **[3 pts]** Give the ***recurrence*** for Quicksort (parallel sort & sequential partition) – best case span. (Note: We are NOT asking for the closed form.) **For any credit, explain all parts of your answer briefly**.

b) **[3 pts]** Give the ***recurrence*** for SEQUENTIAL Mergesort – worst case running time. (Note: We are NOT asking for the closed form.) **For any credit, explain all parts of your answer briefly**.

c) **[2 pts]** Give a tight Big-O bound for the running time of **selection sort** if it *happened to be given an array that was already sorted from smallest to largest.* For any credit explain your answer.

Running time:

7) (Continued)

d) **[2 pts]** Give a tight Big-O bound for the running time of **mergesort** if it *happened to be given an array that was already sorted from smallest to largest.* For any credit explain your answer.

Running time:

e) **[3 pts]** Is bucket sort in-place? YES NO

Explain your answer. Be specific – **refer to the bucket sort algorithm in particular**, do not just give a definition of in-place.

f) **[3 pts]** Is radix sort stable? YES NO

Explain your answer. Be specific – **refer to the radix sort algorithm in particular**, do not just give a definition of stable.

8) **[10 points total] P, NP, NP-Complete**

a) **[1 pt]** “NP” stands for ____

b) **[2 pts]** What does it mean for a problem to be in NP?

c) **[2 pts]** What should you do if you **<u>suspect</u>** (but are not sure) a problem you are given is NP-complete, and you need an answer to the problem in a reasonable amount of time?

d) **[5 pts]** For the following problems, circle **<u>ALL</u>** the sets each problem belongs to:

Finding a cycle that visits each vertex in a graph exactly once	NP-complete	P	NP	None of these
Determining if an array is sorted.	NP-complete	P	NP	None of these
Finding the shortest path from one vertex to all other vertices in the graph. The graph is directed and weighted.	NP-complete	P	NP	None of these
Finding a path in a weighted graph that begins and ends at the same vertex, and visits every vertex exactly once. The path must have a total cost < k.	NP-complete	P	NP	None of these
Determining if a chess move is the best move on an N x N board	NP-complete	P	NP	None of these