CSE 332 Winter 2023 Final Exam

Name: _____

Email address (UWNetID): _____

Instructions:

- The allotted time is 1 hour and 50 minutes.
- Please do not turn the page until the staff says to do so.
- This is a closed-book and closed-notes exam. You are NOT permitted to access electronic devices including calculators.
- Read the directions carefully, especially for problems that require you to show work or provide an explanation.
- When provided, write your answers in the box or on the line provided.
- Unless otherwise noted, every time we ask for an O, Ω, or Θ bound, it must be simplified and tight.
- For answers that involve bubbling in a ◯ or ☐, make sure to fill in the shape completely.
- If you run out of room on a page, indicate where the answer continues. Try to avoid writing on the very edges of the pages: we scan your exams and edges often get cropped off.
- A formula sheet has been included at the end of the exam.

Advice

- If you feel like you're stuck on a problem, you may want to skip it and come back at the end if you have time.
- Look at the question titles on the cover page to see if you want to start somewhere other than problem 1.
- **Relax and take a few deep breaths. You've got this! :-).**

Total: 96 points

Q1: Short-answer questions (10 pts)

- For questions asking you about runtime, give a simplified, tight Big-O bound. This means that, for example, $O(5n^2 + 7n + 3)$ (not simplified) or $O(2^{n!})$ (not tight enough) are unlikely to get points. Unless otherwise specified, all logs are base 2.
- For questions with a mathematical answer, you may leave you answer as an unsimplified formula (e.g. $7 \cdot 10^3$).

We will only grade what is in the provided answer box.

a. Give a **best-case** runtime to find the second largest value in an AVL tree with **N** elements.

b. Given a B-tree of height **3**, with M = **5** and L = **4**, what is the **MINIMUM** possible total number of key-value pairs in the entire tree?

c. Give a simplified, tight big-O bound for: $f(N) = N(\log(N^2)) + N^2(\log N)$

d. Give the **worst-case** runtime of a decreaseKey operation on a Binary Min Heap containing N values.

e. Give the **worst-case** runtime of a pre-order traversal on a Binary Search Tree containing N values

Q2: More Short-answer questions (10 pts)

(Same instructions as for Q1)

a. **True** or **False**: Dijkstra's algorithm can find the shortest path on a graph where all edges have positive weights, but the graph contains cycles.

◯ True ◯ False

b.Give the **worst-case** runtime of a breadth-first search on a dense graph containing V vertices. Assume an adjacency list graph representation is used. **Give your answer in terms of V.**

c. Give the **maximum number of edges** in a Minimum Spanning tree of a connected graph containing V vertices and E edges.

d. Give the **best-case** runtime of the sequential Merge Sort algorithm on an array of **N** elements.

e. What fraction of a program must be parallelizable in order to get **4x** speedup on **6** processors?

Q3: Hashing (9 pts)

For questions a) & b), insert **91, 3, 66, 12, 31, 23, 11** into the tables below. 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) [3 pts] **Quadratic Probing Hashtable.**
If any values cannot be inserted, write them here:_____

0	
1	
2	
3	
4	
5	
6	
7	
8	
9	

b) [3 pts] **Double Hashing Hashtable**. You should use the primary hash function: **h(k) = k % 10** and a secondary hash function: **g(k) = 1 + (k % 4)**.
If any values cannot be inserted, write them here:_____

0	
1	
2	
3	
4	
5	
6	
7	
8	
9	

c) [1 pt] What is the load factor for the table in part a)? _____

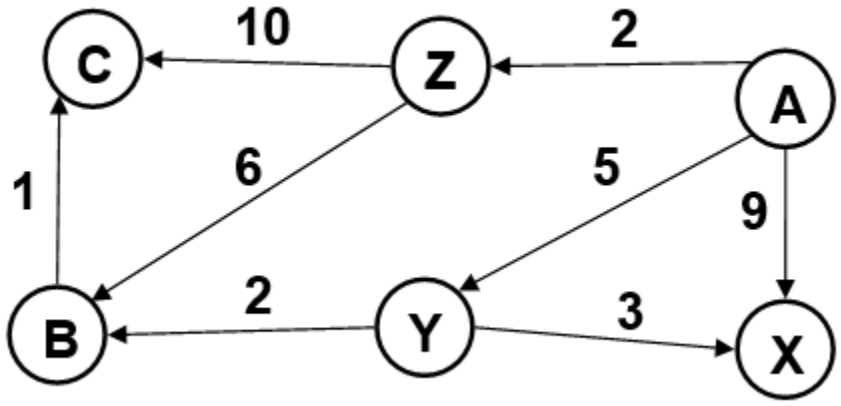
d) [2 pts] What is one advantage and one disadvantage of **quadratic probing** as a collision resolution strategy **compared to linear probing**? Explain each briefly in 1-2 sentences.

One advantage compared to linear probing:

One disadvantage compared to linear probing:

Q4: Graphs (16 pts)

Use the following graph for the problems on this page:



a) [2 pts] List a valid **topological ordering** of the nodes in the graph above (if there are no valid orderings, state why not).

b) [4 pts] Step through Dijkstra's Algorithm to calculate the **single source shortest path from A** to every other vertex. Break ties by choosing the lexicographically smallest letter first; ex. if B and C were tied, you would explore B first. *Note that the next question asks you to recall what order vertices were declared known*. Make sure the final distance and predecessor are clear in the table below.

Vertex	Known	Distance	Predecessor
A			------
B			
C			
X			
Y			
Z			

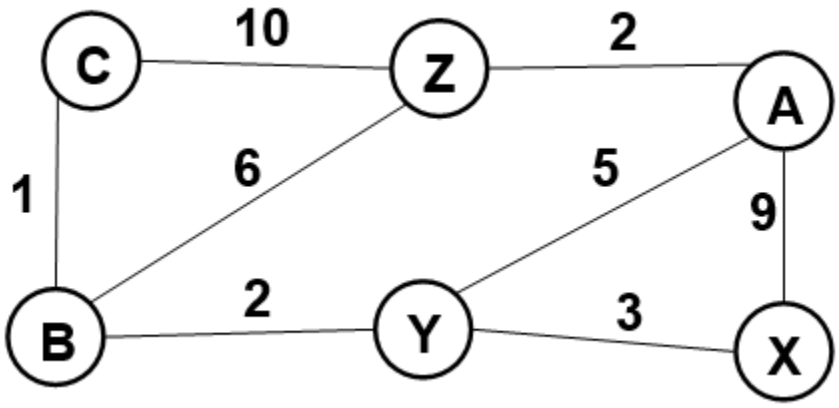
c) [1 pt] In what order would Dijkstra's algorithm mark each node as *known***?**

d) [1 pt] List the **shortest path** from A to C. (Give the actual path **NOT** the cost.)

e) [2 pts] Is this graph **strongly connected**? **Explain your answer in 1-2 sentences for any credit.**

◯ Yes ◯ No

Use the following graph for the problems on this page:



f) [2 pts] What is the **total cost** of a minimum spanning tree in the graph above? ☐

g) [2 pts] ASSUMING the edges above are **unweighted**, give a valid **depth first search** of this graph, **starting at vertex A**, using the non-recursive algorithm described in lecture. **Break ties by choosing the lexicographically smallest letter first**; ex. if B and C were tied, you would add B to the data structure first. You only need to show the final depth first search.

_____, _____, _____, _____, _____, _____

h) [2 pts] ASSUMING the edges above are **unweighted**, give a valid **breadth first search** of this graph, **starting at vertex A**, using the algorithm described in lecture. **Break ties by choosing the lexicographically smallest letter first**; ex. if B and C were tied, you would add B to the data structure first. You only need to show the final breadth first search.

_____, _____, _____, _____, _____, _____

Q5: ForkJoin (14 pts)

In Java using the ForkJoin Framework, write code to solve the following problem:

- **Input**: An array of positive `int`s (does not contain duplicates).
  - You can assume the input array will contain at least two different `int`s.
- **Output**: Returns a **`Pair`** of a) the minimum value and b) its location in the array.

For example, Input array: {4, 6, 1, 2, 3} returns (1, 2), and Input array: {100, 4, 3, 25, 2} returns (2, 4)

- **Do **not** employ a sequential cut-off: **the base case should process one element**.**
  - i.e. you do not **need** to employ a sequential method and you can assume that the code will never process more than one element
- Give a class definition, **`MinLocTask`**, along with any other code or classes needed.
- Fill in the _____ in the function **`findMinLoc`** **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;

public class Pair { // You should use this class
      int min, loc;
      public Pair (int min, int loc) {
            this.min = min;
            this.loc = loc;
      }
}

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

    // Returns the minimum value and its location.
    public static Pair findMinLoc (int[] input) {
       return fjPool.invoke(new MinLocTask(_____________________________________)
    }
}
```

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

Write your class here:

```
public class MinLocTask extends  ____________________________  {
    // Fields go here




    public MinLocTask(_____________________________________________) {




    }
    public _____________compute() {
```

Q6: Concurrency (12 pts)

The following class manages moving charge between your home (solar) battery and your electric car.

```
public class ChargeManager {
    private int solarBatteryCharge = 0;
    private int carBatteryCharge = 0;
    private int chargePerMile = 20;
    private Object solarLock = new Object();
    private Object carLock = new Object();

    public void sunshineAdd(int sun){
        synchronized(solarLock) {
            solarBatteryCharge += sun;
        }
    }

    public void driveCar(int miles){
        synchronized(carLock) {
            if (miles * chargePerMile  > carBatteryCharge)
                throw new CarStrandedException();
            carBatteryCharge -= miles * chargePerMile;
        }
    }

    public void chargeCar(int charge) {
        synchronized(solarLock) {
            synchronized(carLock) {
                if (charge > solarBatteryCharge)
                    throw new NotEnoughSolarChargeException();
                solarBatteryCharge -= charge;
                carBatteryCharge += charge;
            }
        }
    }
}
```

a) [3 pts]Does the **`ChargeManager`** class above have (bubble in all that apply):

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

If there are any problems, describe each problem selected in 1-2 sentences.

b) [3 pts] Does the code above provide any more concurrency than having one lock on the entire **`ChargeManager`** object? **In 1-2 sentences explain why or why not.**

◯ Yes ◯ No

c) [3 pts] You decide to add one more method to the **`ChargeManager`** class:

```
public void chargeSolarFromCar(int charge) {
    synchronized(solarLock) {
        synchronized(carLock) {
            if (charge > carBatteryCharge)
                System.out.println("Warning: Charge values not accurate");
            carBatteryCharge -= charge;
            solarBatteryCharge += charge;
        }
    }
    System.out.println("You lose" + 1.0/chargePerMile + "miles per charge");
}
```

Does adding this method to the **`ChargeManager`** class **cause any new** (bubble in all that apply):

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

If there are any new problems, describe each problem selected in 1-2 sentences.

d) [3 pts] **Instead of** adding in the method above in part c), you add this to the **`ChargeManager`** class:

```
public int reportOverallCharge() {
    return solarBatteryCharge + carBatteryCharge;
}
```

Does adding this method to the **`ChargeManager`** class **cause any new** (bubble in all that apply):

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

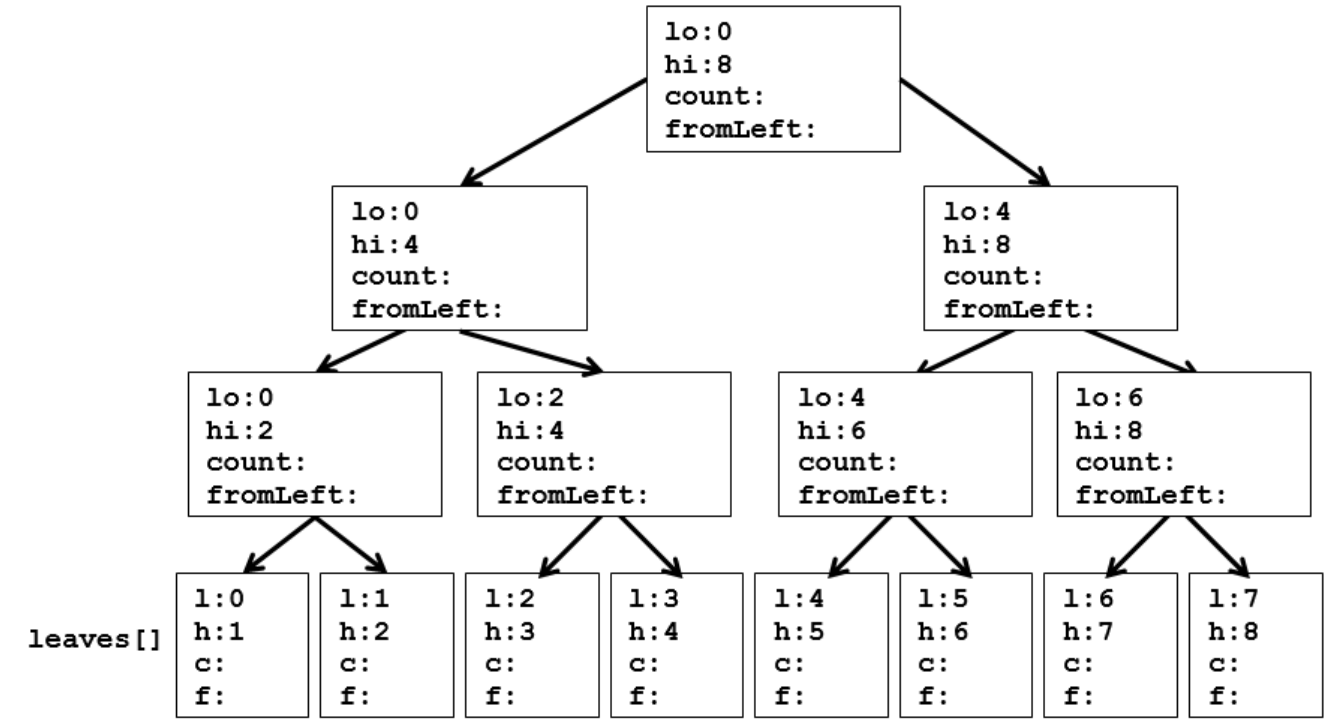
If there are any new problems, describe each problem selected in 1-2 sentences.

Q7: Parallel Prefix (9 pts)

Given the following array and character as inputs, perform the parallel prefix algorithm to fill the output array with the **sum of times the given character (stored in the variable `find`) appears in all of the cells to the left** (including the value contained in that cell) in the **`input`** array. Do not use a sequential cutoff. For example, for
`input = {'a', 'a', 'a', 'b', 'b', 'a', 'a', 'b'}` and **`find = 'b'`**,
`output` should be: **`{0, 0, 0, 1, 2, 2, 2, 3}`**.

a) [5 pts] Fill in the values for **`count`**, **`fromLeft`**, and the **`output`** array in the picture below given the following values for **`input`** and **`n`**. The **`input`** array has already been filled out for you.

```
char[] input = {'n', 'o', 'n', 's', 'e', 'n', 's', 'e'}
char find = 'n'
```



Index	0	1	2	3	4	5	6	7
input	'n'	'o'	'n'	's'	'e'	'n'	's'	'e'
output								

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 on the previous page.

b) [1 pt] Give pseudocode for how you assigned a value to **leaves[i].count**

c) [1 pt] Give code for assigning **p.left.fromLeft.**

p.left.fromLeft =

d) [1 pt] Give code for assigning **p.right.fromLeft.**

p.right.fromLeft =

e) [1 pt] How is **output[i]** computed? Give exact code assuming **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.

output[i] =

Q8: Sorting (10 points)

a) [2 pts] Which of the sorts we discussed in our class would be best to use for this scenario:

Your entire candybar collection expires this year. You need to sort the candy bars by which month they expire in (January-December).

Name of Sort: _____
Justify why your choice is best in 1-2 sentences:

b) [1 pt] True or False: Insertion sort is an **in-place** sort.

◯ True ◯ False
Explain why or why not in 1-2 sentences.

c) [2 pts] Give the *recurrence* for SEQUENTIAL Merge Sort - **best case**

T(n) =

d) [2 pts] Give the *recurrence* for SEQUENTIAL Quicksort - **worst case**.

T(n) =

e) [2 pts] What **parallel algorithm pattern** did we discuss using to parallelize the partition step in Quicksort? **Describe in 1-2 sentences how it was used to do so**.

Name of Pattern: _____

f) [1 pt] What was the new span for just the partitioning step after integrating the technique described in part e).

Q9: P/NP (6 pts)

a) [1 pt] "NP" stands for _____

b) [2 pt] Draw a diagram demonstrating how (we are pretty sure) the sets P, NP, and NP-Complete, overlap / don't overlap with each other.

For the following problems, bubble in **ALL** the sets each problem belongs to:

c) [1 pt] Given a separate-chaining hash table where each bucket is an AVL tree, determine whether the hash table contains the key *k* .

☐ P ☐ NP ☐ NP-complete ☐ none of these

d) [1 pt] Given a connected, undirected graph, determine if it has a path that starts and ends at the same vertex and goes through each **vertex** exactly once.

☐ P ☐ NP ☐ NP-complete ☐ none of these

e) [1 pt] Given a connected, undirected graph, determine if it has a cycle that goes through each **edge** exactly once.

☐ P ☐ NP ☐ NP-complete ☐ none of these

This is a blank page! Enjoy!