CSE 332 Winter 2025 Final Exam

Your
Seat Number: _____

Name: _____

UW NetID: _____(@uw.edu)

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: 93 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.

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

a. **True** or **False**: If the best case runtime of a program is $\Omega(log\ N)$, then the worst case runtime is $\Omega(log\ N)$.

◯ True ◯ False

b. Give a simplified, tight big-O bound for: T(N) = 2*T(N/2) + 1, given T(1) = 1

O( )

c. **True** or **False**: Suppose **`arr`** is a zero-indexed binary min heap of size N. If i, j < N are non-negative integers, such that j = 4 * i + 3, then **`arr[i] <= arr[j]`**.

◯ True ◯ False

d. Give the **worst case** runtime to determine that a key does NOT exist in an AVL tree containing N keys.

O( )

e. What is the maximum number of nodes in a binary search tree with height 7? (Remember: A single node is a tree of height 0.) **Note: we are looking for the exact number here.** For this question partial credit will not be given for formulas or anything other than the actual number - check your work!

Q2: More Short-answer questions (10 pts)

(Same instructions as for Q1)

a. Give the exact number for the minimum number of edges in a complete undirected graph without self-loops containing **7 vertices**. **Note: we are looking for the exact number here.** For this question partial credit will not be given for formulas or anything other than the actual number - check your work!

b**. True** or **False**: In Java fork-join programming, if some data is thread-local, there can never be a data race on that data.

◯ True ◯ False

c. What is the **best case** runtime of selection sort of an array of N unique elements (no duplicates).

O( )

d. What is the **worst case** runtime of a delete operation on a hash table containing N elements, using separate chaining, where each bucket is a sorted linked list. Assume the element is present in the hash table. The load factor of this table is 6.

O( )

e. What fraction of a program must be parallelizable in order to get **3x** speedup on **6** processors? Give your answer as a fraction.

Q3: Hashing (8 pts)

a) [3 pts] **Quadratic Probing Hashtable**. Insert **1, 5, 11, 25, 40, 50, 21, 37** into the table below. 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. Assume no re-sizing occurs during these insertions.

If any values cannot be inserted, write them here:_____

0	
1	
2	
3	
4	
5	
6	
7	
8	
9	

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

c) [2 pt] After completing the insertions above, if you use the method described in lecture to delete 1 from the hash table, what would be the index of 11 after deleting 1?

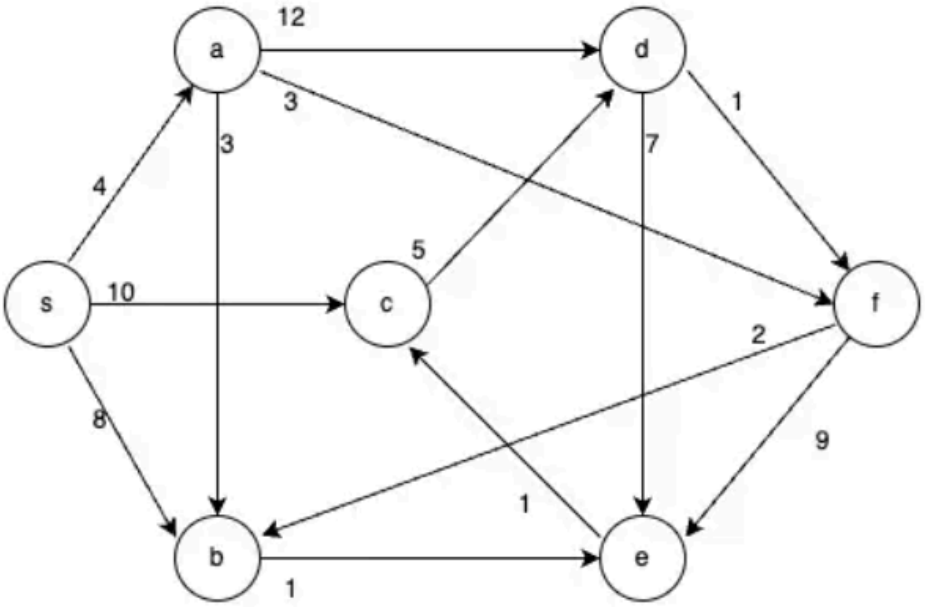
Index of 11:_____

Briefly describe how 1 would be deleted using the method described in lecture (1-2 sentences):

d) [2 pts] Jacklyn is implementing a hash table with Double Hashing. She decided to choose the secondary hash function to be identical to the primary hash function. When using her hash table she encountered an infinite loop on the second insertion to the hash table. **In 2-3 sentences, explain why she might be getting an infinite loop on the second insertion?**

Q4: Graphs (16 pts)

Use the following graph for the problems on this page:



a) [4 pts] Step through Dijkstra's Algorithm to calculate the **single source shortest path from s** 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.

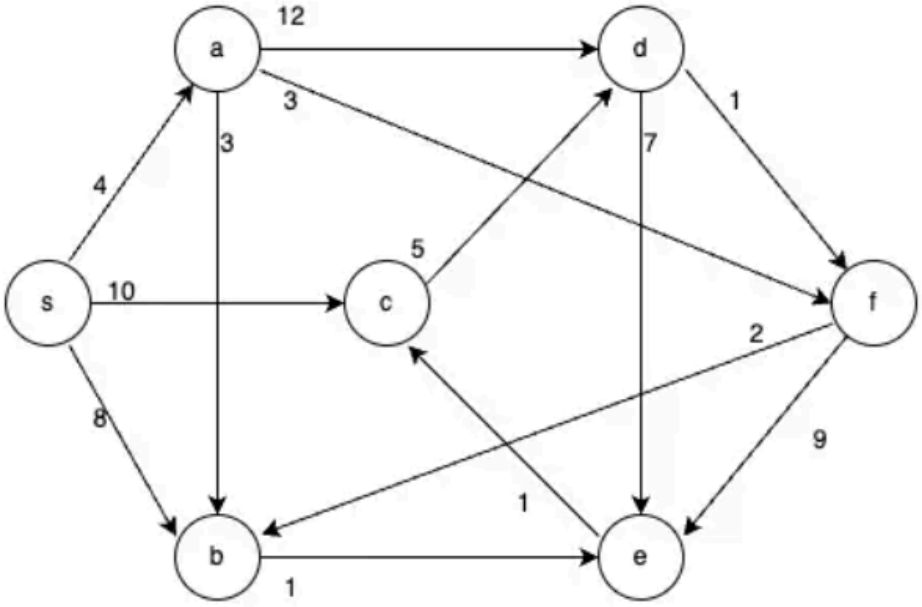
Vertex	Known	Distance	Predecessor
s			------
a			
b			
c			
d			
e			
f			

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

_____, _____, _____, _____, _____, _____, _____

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

Q4 continued: (Copy of graph from previous page)



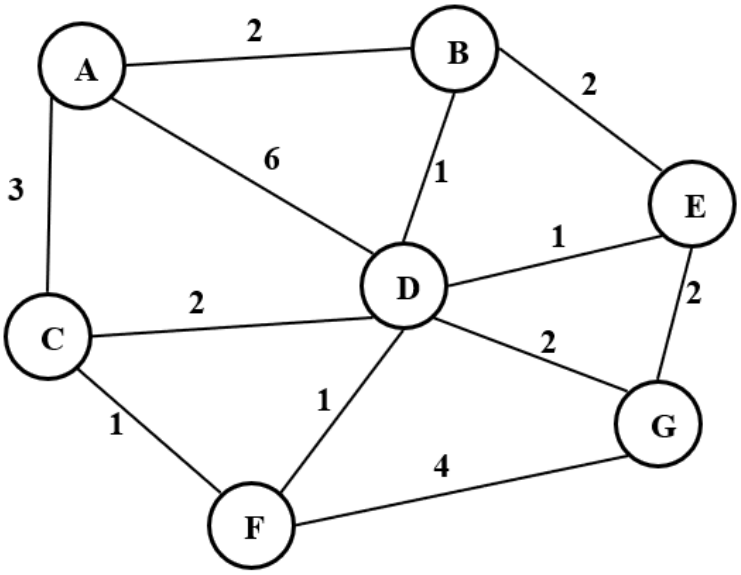
d) [2 pts] List a valid **topological ordering** of the nodes in the graph above. If there are no valid orderings, state why not.

_____, _____, _____, _____, _____, _____, _____

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

◯ Yes ◯ No

Q4 continued: Use the following graph for the problems on this page:



f) [2 pts] If we run Kruskal's algorithm, which of the following edges could be **the last edge added** to the Minimum Spanning Tree? **Bubble in the box for all that apply.** Assume any ties are broken randomly.

☐ AB ☐ AC ☐ AD ☐ BD ☐ BE ☐ CD
☐ CF ☐ DE ☐ DF ☐ DG ☐ EG ☐ FG

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

h) [2 pts] ASSUMING the edges above are **unweighted**, give a valid **breadth first search** of this graph, **starting at vertex F**, using the algorithm described in lecture. **When adding elements to the data structure, you should break ties by choosing the lexicographically smallest letter first**; ex. if A and B were tied, you would add A 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 non-empty lowercase Strings.
- **Output**: Returns a **`Pair`** of a) the total number of Strings that begin with the letter 'z' b) the length of the longest String starting with the letter 'z'. If no words start with 'z', the length should be zero.

For example, Input array: {"zoo", "zebra", "hope", "a", "zipper"} returns (3, 6) and Input array: {"zany", "is", "the", "zest"} returns (2, 4). Java's **`charAt(int i)`** method for Strings returns the i-th character in a String.

- **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, **`FindzTask`**, along with any other code or classes needed.
- Fill in the _____ in the function **`findzWords`** **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 count, len;
      public Pair (int count, int len) {
            this.count = count;
            this.len = len;
      }
}

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

    // Returns the number of Strings that start with z & length of longest z word.
    public static Pair findzWords(String[] input) {
       return fjPool.invoke(new FindzTask(_____________________________________)
    }
}
```

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

****Don't forget to fill in the blank line above!!!!

Write your class here:

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




    public FindzTask(_____________________________________________) {




    }
    public _____________compute() {
```

Q6: Concurrency (12 pts)

The **`SelfDrivingCar`** class manages the charge and location of a self-driving car. Multiple threads might be accessing the same **`SelfDrivingCar`** object. Code for the entire class shown below.

```
1  public class SelfDrivingCar {
2     private String curLoc = "home";
3     private String nextDest = "work";
4     private int chargeLevel = 100;
5
6
7
8
9     public boolean isCharged() {
10
11       return chargeLevel >= 5;
12
13    }
14
15    public void addCharge(int charge) {
16
17        chargeLevel += charge;
18
19    }
20
21    public void setDest(String newDest) {
22
23       nextDest = newDest;
24
25    }
26 }
```

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

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

Give an explanation for each box you checked above (1-2 sentences each). Refer to line numbers in your explanation. Be specific!

b) [3 pts] We now add this method to the **`SelfDrivingCar`** class:

```
27 public boolean driveToDest() {
28
29    if (!isCharged() || curLoc == nextDest) {
30
31       return false;
32
33    }
34
35    curLoc = nextDest;
36
37    chargeLevel = chargeLevel - 5;
38
39    System.out.println("Driving to:" + nextDest + " Charge:" + chargeLevel);
40
41    return true;
42
43 }
```

Does adding this method to the **`SelfDrivingCar`** 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, give an explanation for each box you checked above (1-2 sentences each). Refer to line numbers in your explanation. Be specific!

c) [6 pts] Modify the **code above in part b) and on the previous page** (draw arrows if needed) to ***allow the most concurrent access*** and to avoid all of the potential problems listed above. **For full credit you must allow the most concurrent access possible without introducing any errors or extra locks.** Create locks as needed. Use any reasonable names for the locking methods you call. **DO NOT use `synchronized`.** You should create re-entrant lock objects as follows:

```
ReentrantLock lock = new ReentrantLock();
```

Q7: Parallel Prefix (9 pts)

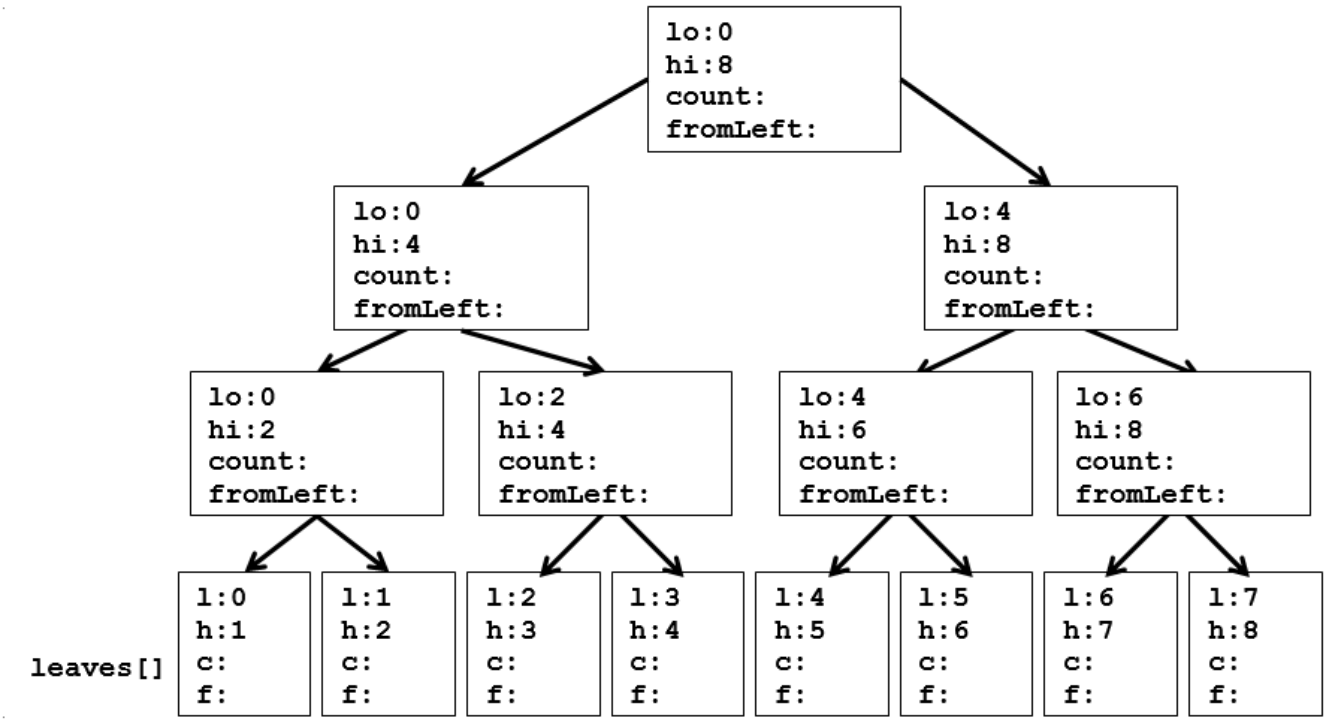
Given the following array and integer as inputs, perform the parallel prefix algorithm to fill the output array with the **count of times a number that is a multiple of the given integer (stored in the variable `mult`) appears in all of the cells to the left** in the `input` array (including the value contained in that cell). Do not use a sequential cutoff.

For example, for `input = {2, 12, 7, 3, 12, 5, -6, 0}` and `mult = 3`,

`output` should be: `{0, 1, 1, 2, 3, 3, 4, 5}`.

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 `mult`. The `input` array has already been filled out for you. Note that problems b-e, on the next page, ask you to give the formulas you used in your calculation

```
int[] input = {3, 8, 2, 6, 100, 4, 17, 4}
int mult = 4
```



Index	0	1	2	3	4	5	6	7
input	3	8	2	6	100	4	17	4
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 (8 pts)

a) [2 pt] We are given an array of size N. Suppose that we are running quicksort on that array and we are choosing the pivot by always choosing the **maximum value** from the elements we are currently partitioning. How many times do we partition the elements? We do not partition on an input size of 1.

$O($ $)$

b) [2 pt] True or False: Radix sort is a **stable** sort.

◯ True ◯ False

Explain why or why not in 1-2 sentences.

c) [2 pts] Give the *recurrence* for Mergesort - **worst case**.

T(n) =

d) [2 pts] In class we discussed a bound on comparison based sorting. What is that bound?

i) Which kind of bound:

◯ Θ ◯ Ω ◯ O

ii) What is the bound?

Q9: Pre-Midterm Medley (6 pts):

a) [3 pts] Give a base case and a recurrence for the runtime of the following function. Use variables appropriately for constants (e.g. c_1, c_2, etc.) in your recurrence (you do not need to attempt to count the exact number of operations). **YOU DO NOT NEED TO SOLVE** this recurrence.

```
int sun(int n) {
    if (n < 2) {
        return n;
    } else {
        for (int i = n; i > 0; i--) {
            print i;
        }
        return sun(n / 2) + sun(n / 3);
    }
}
```

$T(n) =$ _____ For $n < 2$

$T(n) =$ _____ For $n \geq 2$

Yipee!!!! YOU DO NOT NEED TO SOLVE *this* recurrence…

b) [3 pts] Describe the worst-case running time for the following pseudocode function in Big-O notation in terms of the variable n. Your answer **MUST** be tight and simplified. **You do not have to show work or justify your answers for this problem.**

```
int fun(int n) {
      int count = 0;
      for (int k = n; k > 0; k = k/2) {
            for (int j = n; j > 0; j--) {
                  count++;
            }
      }
      return count
  }
```

O( )

This is a blank page! Enjoy!