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.

<table>
<thead>
<tr>
<th>Question</th>
<th>Max Points</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>13</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>100</td>
<td></td>
</tr>
</tbody>
</table>
1) [11 points total] Hash Tables
For a) and b) below, insert the following elements in this order: 19, 48, 8, 27, 97, 7. For each table, TableSize = 10, and you should use the primary hash function \( h(k) = k \mod 10 \). If an item cannot be inserted into the table, please indicate this and continue inserting the remaining values.

a) Separate chaining hash table – use a sorted linked list for each bucket where the values are ordered by increasing value

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

b) Quadratic probing hash table

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

c) What is the load factor in Table a)?

d) Would you implement lazy deletion on Table a)? In 1-2 sentences describe why or why not. Be specific.

e) In 1-2 sentences (or pseudo code) describe how you would implement re-hashing on Table b). Be specific.

f) What is the big-O worst case runtime of an Insert in a hash table like Table a) containing N elements? _____

g) What is the big-O worst case runtime of determining what the maximum value is in a hash table like Table b) containing N elements? _____
2) [10 points total] Graphs!
   a) [2 points] What is the big-O running time of Prim’s algorithm (assuming an adjacency list representation) if a priority queue is used?

   b) [2 points] Give a Minimum Spanning Tree (MST) of the graph below, by highlighting the edges that would be part of the MST.

   ![Graph with edges labeled with weights](image)

   c) [4 points] Kruskal’s
   (i) What is the worst case running time of Kruskal’s algorithm as described in lecture (assuming an adjacency list representation is used)?

   (ii) You try using a new implementation of union-find that claims to have better data locality. find() in this new implementation has a worst case running time of O(V^2) and union() has a running time of O(V). What is the worst case running time of your modified Kruskal’s algorithm that uses this new implementation of union-find?

   d) [2 points] What is the worst case running time to determine whether an edge exists from vertex x to vertex y.

   (i) Given an adjacency matrix representation:

   (ii) Given an adjacency list representation:
3) [9 points total] More Graphs! Use the following graph for this problem:

![Graph Image]

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

b) [3 points] In lecture we described an optimization to the topological sorting algorithm that used a queue. Your partner proposes that you use a priority queue instead of a FIFO queue. What would be the worst case running time of this new version of topological sort (assuming an adjacency list representation of the graph is used)? **For any credit, briefly describe your answer with pseudo code or in a couple of sentences.**

c) [2 points] **ASSUMING the edges above are undirected**, give a valid breadth first search of this graph, starting at vertex A, **using the algorithm described in lecture.**

d) [2 points] **ASSUMING the edges above are undirected**, give a valid depth first search of this graph, starting at vertex A, **using the non-recursive algorithm described in lecture.**
4) [10 points] Parallel Prefix Sum of Positives:

a) Given the following array as input, perform the parallel prefix algorithm to fill the output array with the sum of only the positive values contained in all of the cells to the left (including the value contained in that cell) in the input array. Negative values in the input array should not contribute to the sum. Fill in the values for pSum and fromLeft 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  -1  5  -5 -2  2  -6  9  -3
Output 0  5  5  5  7  7  16  16
```

b) How is the fromLeft value computed for the left and right children of a node in the tree. Give a formula where p is a reference to the current tree node.

\[ p \text{. left. fromLeft} = \]

\[ p \text{. right. fromLeft} = \]

c) How is output[i] computed? Give a formula 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.

\[ \text{output[i]} = \]
5) [14 points] In Java using the ForkJoin Framework, write code to solve the following problem:

• **Input**: An array of ints
• **Output**: the array index of the rightmost number greater than zero in the Input array.
For example, if input array is {-3, 2, 5, -2, 7, 0, -4}, the output would be 4, since that is the index of the value 7. If there are no values greater than 0 in the array then -1 should be returned.

• 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 element.)

• Give a class definition, RightmostPosTask, along with any other code or classes needed.

• Fill in the function findRightmostPos below.

You may not use any global data structures or synchronization primitives (locks).

```java
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 index of the rightmost positive value
    // in the array input. Returns -1 if no values > 0 in input.
    public static int findRightmostPos (int[] input) {
        // Please fill in the function above and write your class on the next page.
    }
}
```
5) (Continued) Write your class on this page.
6) [13 points] Concurrency: The following class implements a Bank account containing both a savings and a checking balance.

```java
public class BankAccount {
    private int savings;
    private int checking;
    private Object savLock = new Object();
    private Object chkLock = new Object();

    void depositToSavings(int amount){
        synchronized(savLock) {
            savings += amount;
        }
    }

    void withdrawFromChecking(int amount){
        if (amount > checking)
            throw new WithdrawTooLargeException();
        checking -= amount;
    }

    void transferSavingsToChecking(int amount) {
        synchronized(savLock) {
            synchronized(chkLock) {
                if (amount > savings)
                    throw new WithdrawTooLargeException();
                savings -= amount;
                checking += amount;
            }
        }
    }
}
```

a) Does the BankAccount class above have (circle all that apply):

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

If there are any problems, describe them in 1-2 sentences.

b) Does the code above provide any more concurrency than having one lock on the entire BankAccount object? In 1-2 sentences explain why or why not.
c) You decide to add one more method to the BankAccount class:

```java
void transferCheckingToSavings(int amount) {
    synchronized(savLock) {
        synchronized(chkLock) {
            if (amount > checking)
                throw new WithdrawTooLargeException();
            checking -= amount;
            savings += amount;
        }
    }
}
```

Does adding this method to the BankAccount class **cause any new** (circle all that apply):

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

If there are any problems, describe them in 1-2 sentences.

d) **Circle** the critical section guarded by chkLock in the method above in part c).

e) **Instead of** adding in the method above in part c), you add this method to the BankAccount class:

```java
int getOverallBalance() {
    return savings + checking;
}
```

Does adding this method to the BankAccount class **cause any new** (circle all that apply):

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

If there are any problems, describe them in 1-2 sentences.
7) [6 points] Speedup

What fraction of a program must be parallelizable in order to get 10x speedup on 20 processors?

You must show your work for any credit. For full credit give your answer as a number or a simplified fraction (not a formula).
8) [16 points] Sorting
a) [3 points] Give the **recurrence** for Quicksort (parallel sort & sequential partition) – best case span: (Note: We are NOT asking for the closed form.)

b) [5 points] Give the big-O runtimes requested below. For parallel sorts, give the span.

_________ A) Selection Sort – best case

_________ B) Quicksort (sequential) – best case

_________ C) Insertion Sort – best case

_________ D) Quicksort (parallel sort & parallel partition) – worst case span

_________ E) Quicksort (sequential) – worst case

c) [5 points] Fill in the blanks.

In class we discussed a Ω (_____________) bound on __________________________ sorting.

Yet we came up with other sorts like __________________________ (name of sorting algorithm) that had better worst case running times of Θ(______________).

Describe in 1-2 sentences why it was possible to come up with these faster algorithms.

d) [1 point] Is radix sort in-place?

YES NO

e) [2 points] In 1-2 sentences, describe what it means for a sort to be in-place?
b) [2 points] What should you do if you suspect (but are not sure) a problem you are given is NP-complete?

c) [5 points] For the following problems, circle ALL the sets they belong to:

Finding the shortest path from one vertex to every other vertex in a directed weighted graph  
NP-complete  
P  
NP  
None of these

Finding a cycle that visits each edge in a graph exactly once  
NP-complete  
P  
NP  
None of these

Determining if a program will ever halt  
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

Finding a cycle in a weighted graph that visits every vertex exactly once and has a total cost < k.  
NP-complete  
P  
NP  
None of these

d) [1 point] If there exists a polynomial time algorithm to solve SAT, then there exists a polynomial time algorithm to solve Hamiltonian Circuit.

TRUE  FALSE

e) [1 point] If there exists a polynomial time algorithm to solve Euler Circuit then any NP-complete problem can be solved by some polynomial time algorithm.

TRUE  FALSE