Name: _____Sample Solution_____

Email address (UWNetID): _____

CSE 332 Autumn 2017 Final Exam

(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.

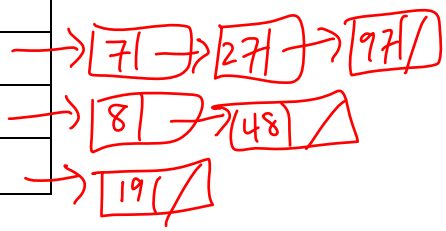
Question	Max Points	Score
1	11	
2	10	
3	9	
4	10	
5	14	
6	13	
7	6	
8	16	
9	11	
Total	100	

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%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**

0	
1	
2	
3	
4	
5	
6	
7	
8	
9	



b) Quadratic probing hash table

0	
1	97
2	8
3	
4	
5	
6	7
7	27
8	48
9	19

c) What is the load factor in Table a)? **0.6**

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

No. Deletion is equivalent in cost to a find operation (once found, just remove the node from the linked list). So you do not really gain much by implementing lazy deletion here – it will merely leave nodes marked deleted in the table, causing other finds/inserts/deletes to have to traverse over them. There could be a small advantage to leaving the node around if it was likely to be inserted again soon, but the advantages of this is probably outweighed by the disadvantages.

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

Two ways you could trigger rehashing are when the load factor on the table gets to be > 0.5 or when an insertion fails. To re-hash, first create a new hash table roughly 2x as large as this one (and a prime number). Traverse the original table from top to bottom, for every value encountered, calculate the hash function on the key and insert it into the new table (mod-ing by the new tablesize).

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

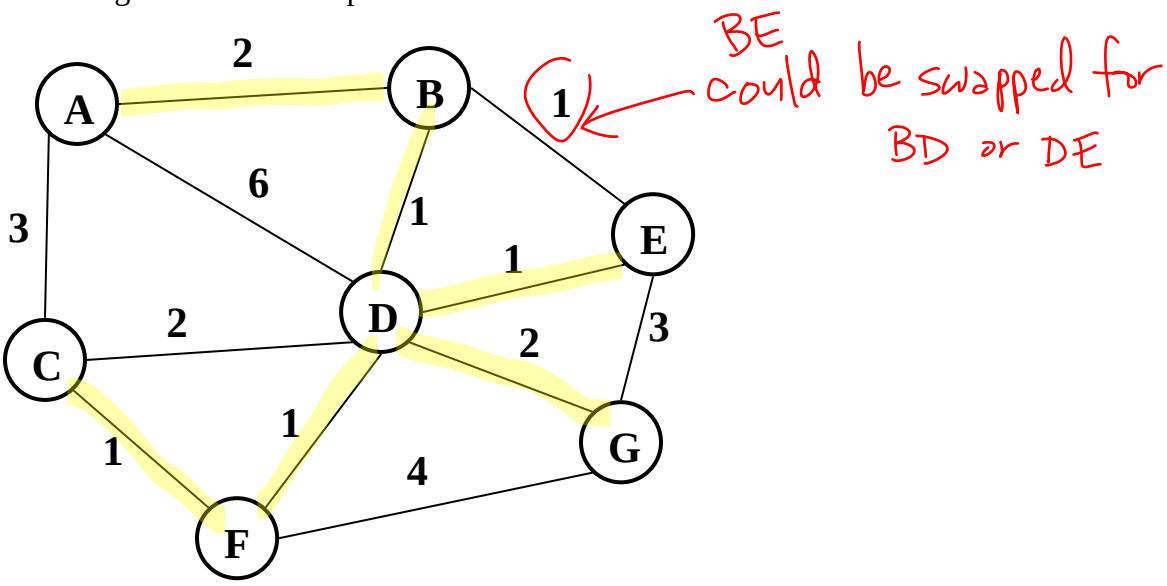
g) What is the big-O worst case runtime of determining what the maximum value is in a hash table like Table b) containing N elements? **_O(N)_**

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?

O(ElogV)

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



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)?

O(E log E) or O(E log V)

(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?

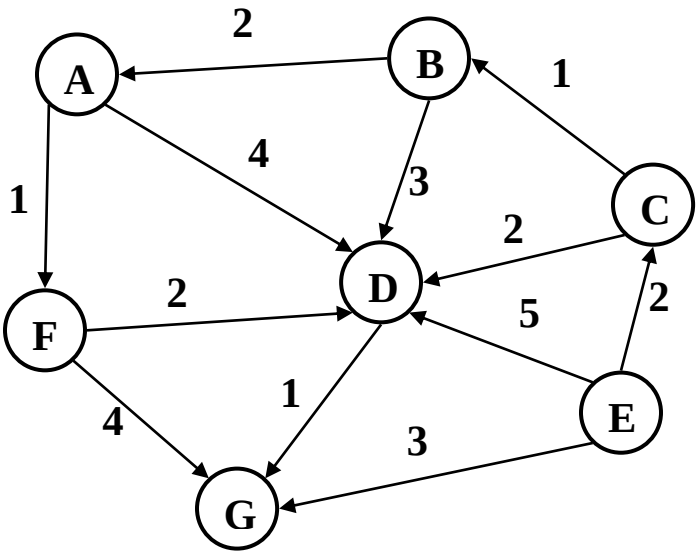
O(E log E + 2E *V^2 +V * V) or $O(E*V^2)$

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: **O(1)**

(ii) Given an adjacency list representation: **O(V) or O(d) where d is out-degree of vertex x**

3) [9 points total] More Graphs! Use the following graph for this problem:



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

E C B A F D G

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.**

```
calculateIndegreeOfAllVertices();   // O(E + V)
buildheapOfVertices();              // O(V)
for (ctr=0; ctr < numVertices; ctr++){ // V times
      v = deletemin();              // O (log V)
      put v next in output          // O(1)
      for each w adjacent to v {    // E times total
            decreasekey(w, 1);      // O(log V)
      }
}  // So we get: O( E + V + V + V log V + E log V) → O(E log V)
```

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

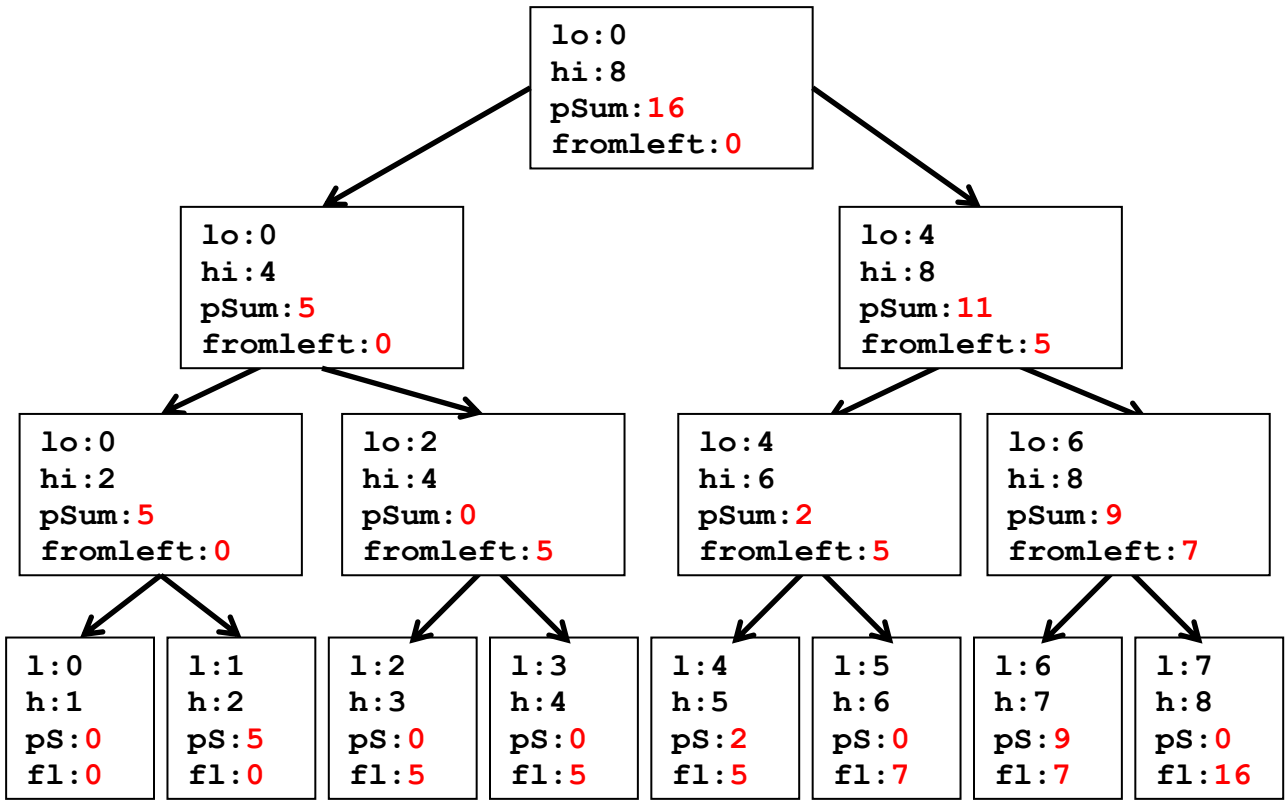
One possibility: **A B D F C E G**

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

One possibility: **A B C E G F D**

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.left.fromLeft` = **`p.fromLeft`**

`p.right.fromLeft` = **`p.fromLeft + p.left.pSum`**

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.

`output[i] = leaves[i].pSum + leaves[i].fromLeft`

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).

```
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) {

          return fjPool.invoke(
                        new RightmostPosTask(input, 0, input.length));
      }
}
```

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

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

```
public class RightmostPosTask extends RecursiveTask<Integer> {
   int[] array;
   int lo;
   int hi;


   public RightmostPosTask(int[] array, int lo, int hi) {
      this.lo = lo;
      this.hi = hi;
      this.array = array;
   }

   protected Integer compute() {
      if (hi - lo < 2) {
         if (array[lo] > 0) {
           return lo;
         } else {
           return -1;
         }
      } else {
         int mid = lo + (hi - lo) / 2;
         RightmostPosTask left = new RightmostPosTask (array,
lo, mid);
         RightmostPosTask right = new RightmostPosTask (array,
mid, hi);

         left.fork();
         int rightResult = right.compute();
         int leftResult = left.join();

         if (rightResult > leftResult) {
            return rightResult;
         } else {
            return leftResult;
         }
      }
   }
}
```

6) [13 points] Concurrency: The following class implements a Bank account containing both a savings and a checking balance.

```
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){
            synchronized(chkLock) {
                  if (amount > checking)
                        throw new WithdawTooLargeException();
                  checking -= amount;
            }
      }

      void transferSavingsToChecking(int amount) {
            synchronized(savLock) {
                  synchronized(chkLock) {
                        if (amount > savings)
                              throw new WithdawTooLargeException();
                        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.**

Yes, a thread could be calling `withdrawFromChecking` at the same time as another thread is calling `depositToSavings`.

c) You decide to add one more method to the `BankAccount` class:

```
void transferCheckingToSavings(int amount) {
      synchronized(savLock) {
            synchronized(chkLock) {
                  if (amount > checking)
                        throw new WithdawTooLargeException();
                  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:

```
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.

There is a data race on the savings and checking fields. A thread could be reading `savings` and `checking` inside of `getOverallBalance` while at the same time another thread is writing one or more of those values in any of the other synchronized methods. A data race is a type of race condition.

7) [6 points] Speedup

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

<u>You must show your work for any credit</u>. For full credit give your answer as a number or a simplified fraction (not a formula).

$T_p = S + \left(\frac{1-S}{P}\right)$

$\frac{T_1}{T_{20}} = 10 = \frac{1}{S + \left(\frac{1-S}{20}\right)}$

$10\left(S + \left(\frac{1-S}{20}\right)\right) = 1$

$S + \frac{1-S}{20} = \frac{1}{10}$

$\frac{20 \cdot S + 1 - S}{20} = \frac{1}{10}$

$\frac{19 \cdot S + 1}{20} = \frac{1}{10}$

$\frac{19 \cdot S + 1}{2} = 1$

$2 = 19 \cdot S + 1$

$1 = 19 \cdot S$

$S = \frac{1}{19}$

$\frac{18}{19}$ of the program must be parallelizable

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.)

T(n) = O(n) + T(n/2)

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

__**$O(n^2)$**____ A) Selection Sort – best case

__**O(nlogn)**_ B) Quicksort (sequential) – best case

__**O(n)**____ C) Insertion Sort – best case

__**O(nlogn)**_ D) Quicksort (parallel sort & parallel partition) – worst case span

__**$O(n^2)$**____ E) Quicksort (sequential) – worst case

c) **[5 points]** Fill in the blanks.

In class we discussed a Ω (__**nlogn**_____) bound on __**comparison-based**_____ sorting. Yet we came up with other sorts like __**radix sort**___ (name of sorting algorithm) that had better worst case running times of Θ(_____**n**_____).

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

Our new algorithms made progress without comparing one element to another. We also made assumptions about the values we were sorting. Knowing the range of possible values allowed us create buckets (or a number of buckets and a number of passes over those buckets) that gave us a O(n) bound.

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?

In place means using no more than a constant amount more space. A few extra variables may be used, but O(n) extra space may not be used.

9) **[11 points] P, NP, NP-Complete**

a) **[2 points]** "NP" stands for ___**Non-deterministic polynomial** _____

b) **[2 points]** What should you do if you **suspect** (but are not sure) a problem you are given is NP-complete?

First try to establish that it is NP-complete. Part of this includes doing a reduction from a known NP-complete problem to your problem in polynomial time. (Another part includes showing that your problem is in NP, although that alone is not sufficient.)

AFTER you have shown that your problem is NP-complete, you can quit trying to find a polynomial time algorithm and instead use any of the techniques we use for NP-complete problems.

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**