Name: _____

Email address (UWNetID): _____

CSE 332 Winter Final Exam Review

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.

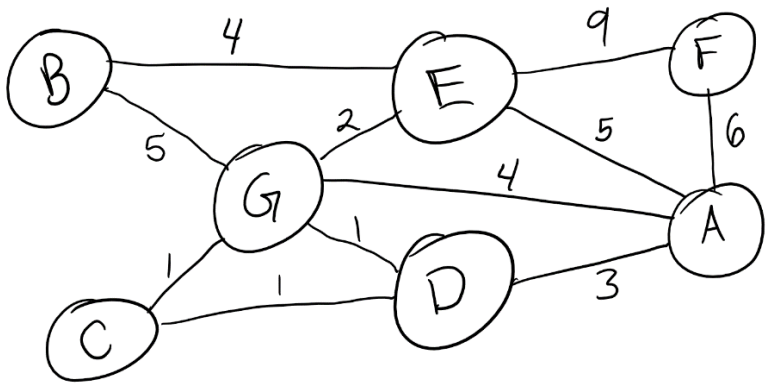
You have 1 hour and 50 minutes, work quickly and good luck!

Total: Time: 1 hr and 50 minutes.

Question	Exam
2	18 Winter
3	18 Winter
5	18 Winter
7	18 Autumn
8	18 Autumn
9	18 Winter

2) [9 points total] Graphs!

a) **[6 points]** Find a minimum spanning tree with Prim's algorithm **using vertex G as the starting node** (mark, circle, or highlight edges below to indicate they are in your minimum spanning tree). You **must show your steps** in the table below for full credit. Show your steps by crossing through values that are replaced by a new value. Break ties by choosing the lowest 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*.



	Cost	Prev	Known?
A			
B			
C			
D			
E			
F			
G			

b) **[1 point]** List the order the vertices are added to the known set:

c) **[1 point]** Pick a node you could start at to get a *different minimum spanning tree* than the one you found in part a). Which edge would be in this new tree that is ***not*** in your tree above? ***DO NOT DRAW THE WHOLE TREE.***

Starting node (for example, "Z"): _____ **Edge** (for example, "(X, Y)"): _____

d) **[1 point]** Will Prim's starting at vertex G find a correct minimum spanning tree if the weight of edge (A,F) is set to be -6? (circle one)

YES **NO**

3) [11 points total] More Graphs!

a) [2 points] If you needed to calculate the out-degree of all vertices in a graph, which representation would you prefer (circle one):

adjacency matrix or **adjacency list**

In a couple of sentences describe WHY?

b) [2 points] Give an example of a directed graph with exactly two topological orderings and one node of in-degree zero.

c) [1 **point**] Let G be a connected, undirected, weighted graph. Convert G to a directed graph as follows: replace every undirected edge (u, v) with directed edges (u -> v) and (v -> u). The resulting graph is strongly connected.

TRUE FALSE

d) **[4 points]** What is the worst case running time of Dijkstra's algorithm described in lecture that:

i. Does NOT use a priority queue:

ii. Uses a priority queue:

e) **[2 points]** Give an EXACT number (in terms of V) for:

i. Maximum number of edges in an undirected graph without self-loops:

ii. Minimum number of edges in a weakly connected directed graph

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

• **Input**: An array of positive `ints`
• **Output**: an array of 10 `ints` containing a count of the **ones place digits** of the values in the Input array. The count of digit `i` will be in Output[`i`].
For example, if the input array is {2007, 13, 17, 24, 5, 17, 38, 407, 0, 7, 4, 17}, the output array (always containing exactly 10 `ints`) would be {1, 0, 0, 1, 2, 1, 0, 6, 1, 0}.

- 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, `CountOnesPlaceTask`, **along with any other code or classes needed**.
- Fill in the function `findOnesPlaces` **below.**

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

a) Write the code.
b) Answer this: Is this a **map** or a **reduction (circle one)**? **Why**?

```
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 an array of 10 ints.  Where the ith element
     // contains a count of the number of times i appears in
     // the ones place in the values in input.
     public static int[] findOnesPlaces (int[] input) {
```

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

Don’t forget to answer b) on the previous page!

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

7) [22 points] Code Analysis & Sorting - A friend of yours has collected an array of delicious recipes to try baking. Each line of the recipe contains either an "ingredient" (e.g. `"ingredient: eggs"`) or an actual step in the baking process (e.g. `"step: crack the eggs into the bowl"`). Recipes always have at least one ingredient and at least one step. A sample recipe with a total of 7 lines and 4 steps might look like this:

```
ingredient: butter
ingredient: sugar
step: mix the butter and sugar
ingredient: flour
step: Add flour
step: Mix well
step: Bake
```

Your friend wrote some code to sort the recipes by how many **steps** they have, so that they can try baking the recipes in order of difficulty, from easiest (fewest number of steps) to hardest (largest number of steps). Their (pseudo)-code is shown below:

```
// sorts the provided array of recipes in ascending order of difficulty
void confectionSort(recipes):
    for i from 0 to recipes.length - 1:
        for j from i to recipes.length - 1:
            if isEasier(recipes[j], recipes[i]):
                  swap(recipes[i], recipes[j])

// helper function that returns true iff recipeA has fewer steps than recipeB
boolean isEasier(recipeA, recipeB):
    stepsInA = 0
    stepsInB = 0

    for line in recipeA.lines:
        if line is a step:
            stepsInA++
    for line in recipeB.lines:
        if line is a step:
            stepsInB++
    return stepsInA < stepsInB
```

Your friend wants your help analyzing and improving the code. **For each of the following four options** give the worst case Big-Oh runtime of the `confectionSort` algorithm, in terms of ***R*** (the number of recipes), and ***L*** (the maximum number of lines in any single recipe). Keep all terms of ***R*** and ***L*** in your final answers (e.g. do not assume **R** > ***L***).

a) **ORIGINAL SORT**: [Code shown above]
Running time:

b) CHANGE #1: [Original `isEasier` function, NEW `confectionSort` function that uses a **merge-sort strategy**.]
Running time:

7) (Continued)

c) **CHANGE #2**: [NEW Modified `isEasier` function, Original `confectionSort` function.]
Describe how you would modify the `isEasier` function to improve overall `confectionSort` running time. You are allowed to add additional fields to the `recipe` objects if needed. You do not need to show code.
Description of Change:

Running time:

d) **CHANGE #3**: [Throw everything out and start over!]
After talking some more, your friend mentions that although recipes sometimes contain many ingredients, every recipe has **at most 10 steps**. You are overjoyed! Describe what sorting algorithm your friend **should** be using.
Description of Change/New algorithm:

Running time:

e) Is ORIGINAL `confectionSort` an in-place sort? For any credit, briefly explain why.
YES NO
Explanation:

f) Is ORIGINAL `confectionSort` a stable sort? For any credit, briefly explain why.
YES NO
Explanation:

---------- Below here not related to `confectionSort` ----------

g) **[3 points]** Give the ***recurrence*** for Quicksort (parallel sort & sequential partition) – worst case span: (Note: We are NOT asking for the closed form.)

`T(n) =`

h) **[3 points]** Quicksort's partition step can also be parallelized. **What** is the big-O span of a single parallel partition? For full credit, **explain why** it has that span?

8) [6 points] Speedup

What ***fraction of a program must be parallelizable*** in order to get 5x speedup on 15 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).

9) [6 points] Speedup

Your boss wants 111x speedup on a program of which 9/10 is parallelizable. What do you tell them? At least how many processors would you need?

Justify your answer with a computation. No credit given without an explanation.