Final Project Questions for CSCI 417

Read the instructions carefully:

- There are 8 questions in this document, in no particular order. Some of them are worth 30 points, and some are worth 15 points. **You must attempt either one 30-point question *OR* two 15-point questions**.
- This is **NOT** a group project. Everyone must make an independent submission.
- You can either use Java or Python (for some questions, the choice would be obvious)
- Make sure your program can be run (i.e., it's free from syntax errors) before you submit. Having syntax errors, such as typos in statements, will be penalized an extra 5 points.
- The above point must be observed even if your program has a logical error and doesn't work as expected.
- You can ask me if you run into any difficulties.
- You can submit this until the day of the scheduled final exam (but there is no final exam for this course)

Problems:

---

1. **[30 points]** Write a (preferably abstract) class called Sorting. It should have methods called bubbleSort(), selectionSort(), quickSort() and mergeSort(). Each of these methods, obviously, take an array and sort it using the respective method. There should also be a method called isSorted() that takes an array as input and returns the Boolean value true if it is sorted, and false if not.
   In another class, the main method should create four arrays of 100000 random numbers, sort them using the four methods of the Sorting class, and print the time needed to run.

2. **[15 points]** Write a function to insertionSort an array, and another that takes in an array and Shell sorts it. You can a "sequence" of your choice to reduce the gap between iterations. Your project should also contain a part (main method if using java) that creates two random arrays of size 100000, sorts them by the two methods, and prints the time. [Hint: You may make use of one function within another].

3. **[15 points]** Write a program to simulate the game of Taxman. Taxman is a one-player number game. The player chooses how many numbers (positive integers) are in the game, from 1 to some upper limit. During the course of the game, the player and the computer, hereafter referred to as the Taxman, each accumulates a total. The object of the game is for the player to accumulate a larger total than the Taxman.

   The player's total accumulates simply by selecting one of the numbers left in the game. The taxman then gets all the numbers left in the game that divide evenly into the player's chosen number. Once numbers are used (either by the player or the Taxman), they are removed from the game.

   There is one major restriction on the numbers that the player may select. As in real life, the Taxman must always get something, so the player can never select a number unless at least one proper divisor of that number remains in the game. Once no numbers with divisors remain (at the end of the game), the Taxman gets all the numbers left and the game is over.

   For example, suppose the game is played with 1, 2, 3, 4, 5, and 6. If the greedy player chooses 6, then the Taxman gets all the divisors of 6, namely 1, 2, and 3. Now the only numbers left in the game are 4 and 5. Neither has a divisor left in the game, so the Taxman gets those also and wins 15 to 6. However, if the player is a bit smarter and chooses 5 first, the player gets 5 and the Taxman gets 1. Now the numbers remaining are 2, 3, 4, and 6. The smart player chooses the 4 (before the 6), giving the Taxman 2. Finally the player chooses 6, giving the Taxman 3. This time the player wins 15 to 6.

   You may make the upper limit for a game 100. **Remember, the question is not to design a program that PLAYS the game. The question is to design a program that lets the user play the game and reacts to their moves.**

4. **[30 points]** If you design the game in Question 3 above, and also design a program that **PLAYS** the game, **resulting in a win every time**, you get 30 points. Remember, I'm not asking for a program guaranteed to get the maximum score, but merely guaranteed to get a higher score than the Taxman.

5. **[15 points]** Implement Kruskal's Algorithm for finding the minimum spanning tree (MST) of a graph using the disjoint set implementation discussed in class. The input to this program is a weighted graph stored as an adjacency matrix, and the output is the MST stored as an adjacency matrix.

6. **[30 points]** Closest pairs problem: Given n points on a plane (in the form of x,y coordinates), write an efficient algorithm to find the pair of points that are closest to each other. You can read more about this problem in the attached document.

7. **[30 points]** Floyd's Algorithm to find the shortest path between all pairs in a graph. More about this is given in the attached document. The input to this program is a graph represented by an adjacency matrix showing edge weights, and the output is a matrix showing the shortest distance between every pair of vertices.

8. **[15 points]** Write an efficient algorithm to find the majority element in an array, where the elements of the array can't be sorted (they have no ordering). They can only be compared to each other for equality/inequality). I'm looking for a solution better than $O(n^2)$, and one with 100% certainty (not the guessing solution)