CST 370 – Spring A 2020
Homework 6

Name:____Ivan Alejandre__________________

Class ID: _____8889____________

How to turn in?

- Write your answer to the questions 1 to 6, and submit it on the iLearn. You can submit the file in PDF format. Don't forget to write your name and class ID number at the beginning of the file.
- For Questions 7 and 8, you should submit your C++ source file on the iLearn.
- Thus, you have to submit three files (one PDF file and two C++ source files) on the iLearn.
- Note that the due date is 11:55(PM). This is the iLearn's timestamp, not your submission time. Since there could be a long delay between your computer and iLearn, you should **submit early**.

Note: If it is difficult to draw diagram(s) when doing homework, you can draw them on paper by hand. Then take a picture of the paper and insert it.

1. (5 points) Construct a heap for the list 1, 8, 6, 5, 3, 7, 4. You should assume that the numbers are given in that order and use the bottom-up approach covered this week.

Insert all numbers into tree:

$$\begin{tikzcd}
 & & & 1 \arrow[dll] \arrow[drr] & & & & \\
 & 8 \arrow[dl] \arrow[dr] & & & & 6 \arrow[dl, leftrightarrow] \arrow[dr] & & \text{swap 6 \& 7} \\
5 & & 3 & & 7 & & 4 &
\end{tikzcd}$$

$$\begin{tikzcd}
 & & & 1 \arrow[dll, leftrightarrow] \arrow[drr] & & & & \\
 & 8 \arrow[dl] \arrow[dr] & & & & 7 \arrow[dl] \arrow[dr] & & \text{swap 8 \& 1} \\
5 & & 3 & & 6 & & 4 &
\end{tikzcd}$$

$$\begin{tikzcd}
 & & & 8 \arrow[dll] \arrow[drr] & & & & \\
 & 1 \arrow[dl, leftrightarrow] \arrow[dr] & & & & 7 \arrow[dl] \arrow[dr] & & \text{swap 5 \& 1} \\
5 & & 3 & & 6 & & 4 &
\end{tikzcd}$$

$$\begin{tikzcd}
 & & & 8 \arrow[dll] \arrow[drr] & & & & \\
 & 5 \arrow[dl] \arrow[dr] & & & & 7 \arrow[dl] \arrow[dr] & & \text{complete} \\
1 & & 3 & & 6 & & 4 &
\end{tikzcd}$$

2. (10 points) Assume that you are constructing a heap with five nodes **10**, **20**, **30**, **40**, and **50**. But you don't know the input order of the numbers. Draw all possible different heaps with the five nodes that satisfy the requirement of a heap.

$$\begin{tikzcd}
 & & 50 \arrow[dl] \arrow[dr] & \\
 & 40 \arrow[dl] \arrow[dr] & & 30 \\
10 & & 20 &
\end{tikzcd}$$

$$\begin{tikzcd}
 & & 50 \arrow[dl] \arrow[dr] & \\
 & 40 \arrow[dl] \arrow[dr] & & 20 \\
10 & & 30 &
\end{tikzcd}$$

$$\begin{tikzcd}
 & & 50 \arrow[dl] \arrow[dr] & \\
 & 40 \arrow[dl] \arrow[dr] & & 30 \\
20 & & 10 &
\end{tikzcd}$$

$$\begin{tikzcd}
 & & 50 \arrow[dl] \arrow[dr] & \\
 & 30 \arrow[dl] \arrow[dr] & & 40 \\
20 & & 10 &
\end{tikzcd}$$

$$\begin{tikzcd}
 & & 50 \arrow[dl] \arrow[dr] & \\
 & 40 \arrow[dl] \arrow[dr] & & 20 \\
30 & & 10 &
\end{tikzcd}$$

$$\begin{tikzcd}
 & & 50 \arrow[dl] \arrow[dr] & \\
 & 30 \arrow[dl] \arrow[dr] & & 40 \\
10 & & 20 &
\end{tikzcd}$$

3. (10 points) Consider a problem of scheduling *n* jobs of known durations t_1, t_2, ..., t_n for execution by a single processor. The jobs can be executed in any order, one job at a time. You want to find a schedule that minimizes the total time spent by all the jobs in the system. (The time spent by one job in the system is the sum of the time spent by the job in waiting plus the time spent on its execution.)

(a) Design **a greedy algorithm** (describe in English) for this problem.
First sort the times in ascending order. Then execute the jobs in that order

(b) Does your algorithm **always provide an optimal solution?** (Yes/No). If your answer is "No", present a counterexample.
Yes, it will always be optimal because the greedy algorithm provides the optimal solution.

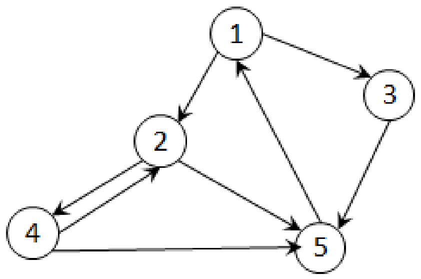
4. (10 points) This is about the Coin-Row problem. Assume that there are six coins like below
5, 4, 1, 8, 10, 6

Fill out the table using dynamic programming method as discussed in the class After that, present the set of coins to be picked and maximum amount. Do you have only one optimal solution?

Index	0	1	2	3	4	5	6
C_i	--	5	4	1	8	10	6
F(i)	0	5	5	6	12	16	18

Solution: F(6) = 18
Coins needed: C_6, C_4, C_2

5. (10 points) Assume that you have the following directed graph. Find the transitive closure of the graph using the algorithm covered in the class. In other words, you have to present $R^{(0)}$, $R^{(1)}$, $R^{(2)}$, $R^{(3)}$, $R^{(4)}$, and $R^{(5)}$ to solve the problem. For your understanding, $R^{(0)}$ is presented below as an example. Thus, you should present the $R^{(1)}$, $R^{(2)}$, $R^{(3)}$, $R^{(4)}$, and $R^{(5)}$.



R(0)

	1	2	3	4	5
1	0	1	1	0	0
2	0	0	0	1	1
3	0	0	0	0	1
4	0	1	0	0	1
5	1	0	0	0	0

R(1)

	1	2	3	4	5
1	0	1	1	0	0
2	0	0	0	1	1
3	0	0	0	0	1
4	0	1	0	0	1
5	1	1	1	0	0

R(2)

	1	2	3	4	5
1	0	1	1	1	1
2	1	0	0	1	1
3	1	0	0	0	1
4	0	1	0	0	1
5	1	1	1	0	0

R(3)

	1	2	3	4	5
1	1	1	1	1	1
2	1	1	1	1	1
3	1	1	1	0	1
4	1	1	1	0	1
5	1	1	1	1	1

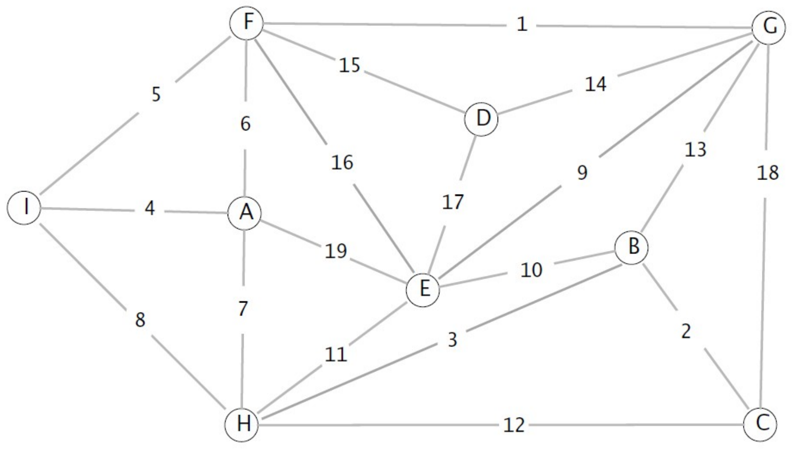
R(4)

	1	2	3	4	5
1	1	1	1	1	1
2	1	1	1	1	1
3	1	1	1	1	1
4	1	1	1	1	1
5	1	1	1	1	1

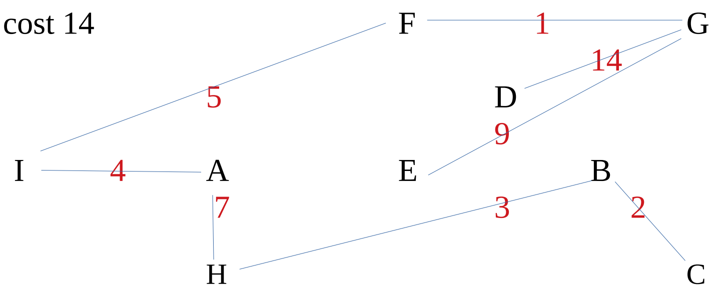
R(5)

	1	2	3	4	5
1	1	1	1	1	1
2	1	1	1	1	1
3	1	1	1	1	1
4	1	1	1	1	1
5	1	1	1	1	1

6. (5 points) Find the minimum spanning tree (MST) to the following graph using the Prim's algorithm as you learned in the class. For the problem, you have to start the algorithm from the vertex I. Then, present the sequence of vertices/edges to be added, indicate the weights for the edges, and draw the result MST clearly.



(1) New Edge: I,A - cost 4
(2) New Edge: I,F - cost 5
(3) New Edge: F,G - cost 1
(4) New Edge: A,H - cost 7
(5) New Edge: H,B - cost 3
(6) New Edge: B,C - cost 2
(7) New Edge: G,E - cost 9
(8) New Edge: G,D - cost 14



7. (25 points) Write a C++ program called **heap.cpp** to conduct heap operations. When the program starts, it should read a set of numbers from a user and store them in an array (starting from index 1) in the order the numbers are entered. After that, your program should display if the array representation is a heap or not. If it's not a heap, your program should construct a heap (heapify) using the bottom-up algorithm. In addition, your program should support 3 operations that a user can select, as shown below.

This is a sample run:

```
Input size: 5
Enter numbers: 20 7 8 1 3
This is a heap.
Select an operation
    1: Insert a number
    2. Delete the max
    3. Heapsort & Quit
1
Enter a number: 5
```

```
Updated heap: 20 7 8 1 3 5

Select an operation
    1: Insert a number
    2. Delete the max
    3. Heapsort & Quit
2
Updated heap: 8 7 5 1 3

Select an operation
    1: Insert a number
    2. Delete the max
    3. Heapsort & Quit
2
Updated heap: 7 3 5 1

Select an operation
    1: Insert a number
    2. Delete the max
    3. Heapsort & Quit
3
Heapsort: 7 5 3 1
Bye!
```

This is another sample run.

```
Input size: 3
Enter numbers: 1 3 7
This is NOT a heap.
Heap constructed: 7 3 1
Select an operation
    1: Insert a number
    2. Delete the max
    3. Heapsort & Quit
1
Enter a number: 100
Updated heap: 100 7 1 3

Select an operation
    1: Insert a number
    2. Delete the max
    3. Heapsort & Quit
3
Heapsort: 100 7 3 1
Bye!
```

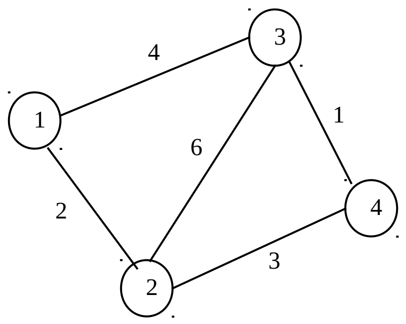
8. (25 points) Write a C++ program called **prim.cpp** that implements the Prim's algorithm to calculate the minimum spanning tree (MST) from an input graph. Your program should ask a user for an input file name that includes the input graph information. Then, your program should present the edges to be added one-by-one to construct the MST. In the problem, you can assume that the maximum number of vertices

(= nodes) in the input graph is less than or equal to 20. You can also assume that the input graph is always connected.

Input file format: This is a sample input file called **`t1.txt`**.

```
4
5
1 2 2
2 3 6
1 3 4
2 4 3
3 4 1
```

The first line (= 4 in the example) indicates that there are four vertices (=nodes) in the graph. The second line (= 5 in the example) presents the number of edges in the graph. The remaining five lines present the edge information (= starting vertex, ending vertex, cost) in the graph. For the homework, you should assume that the first vertex starts from the number 1. Thus, **t1.txt** describes a graph like below:



For the homework, you have to assume that a blank space is used to delimiter the values in the edge information. However, there's no space at the end of each line. If your program does not read the file properly, your program will get no credit. And also, **do not assume that the starting vertex is always 1**. **Your program has to ask the user for the starting vertex.**

This is a sample run

```
Enter a file name: ./t1.txt
Enter the first vertex to start: 1

(1) New edge: 1,2 – cost 2
(2) New edge: 2,4 – cost 3
(3) New edge: 3,4 – cost 1
```

At the result, **the sequence of vertices in an edge is not important.** For example, the following is fine

```
(1) New edge: 1,2 – cost 2
```

OR

```
(1) New edge: 2,1 – cost 2
```