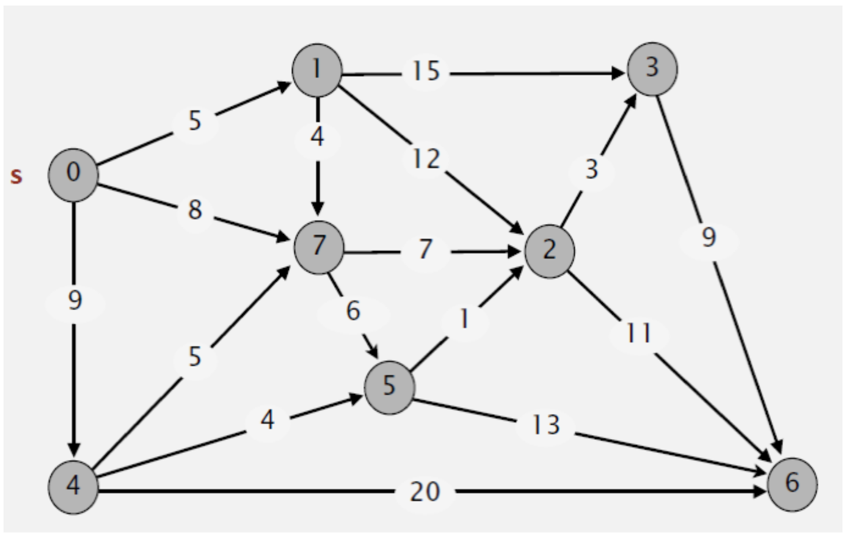
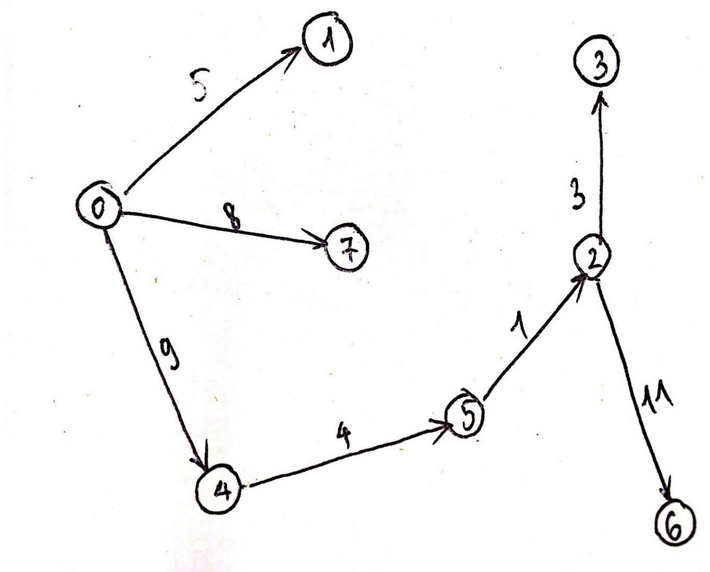
Exercise 5. Answer Sheet

Student's Name: ___Tran Thi Thoa____ Student's ID: _____s1242006_____

Problem 1. *(15 points)* Consider the graph below.



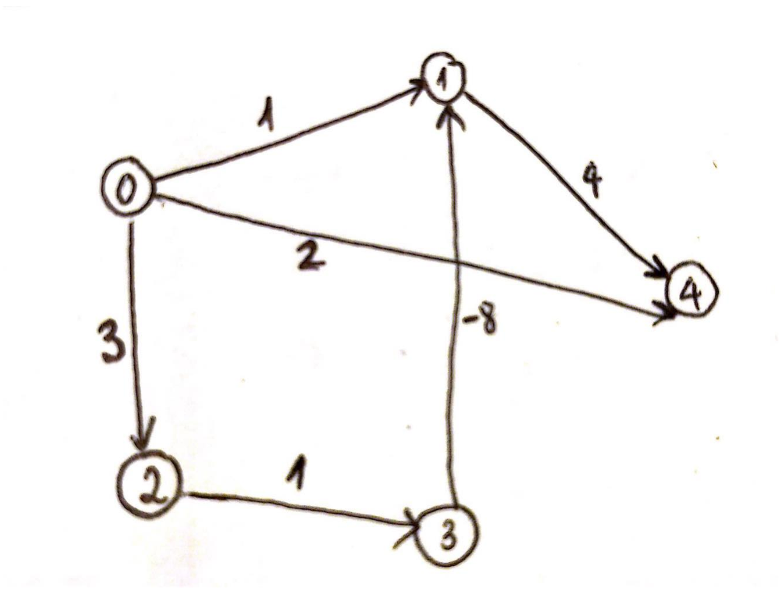
Draw a shortest path spanning tree with root at vertex ***s***. Show the cost (weight) of paths to each vertex.



Problem 2. *(15 points)* Dijksta's algorithm cannot handle negative weights. Show an example and explain what happens.

Dijkstra's algorithm is based on the greedy method. It adds vertices by increasing distance. if all weights are non-negative, adding an edge can never make a path shorter. That's why picking the shortest candidate edge (local optimality) always ends up being correct (global optimality). However, if negative edges exist in the graph, it may takes to the false result.

For example:



For the above graph, we can see that, if using Dijkstra's algorithm to find the shortest path from the vertex 0, we can see that it selects immediately the vertex 1 after that and chooses (0,1) as the shortest path from 0 to 1 with the weight is equal to 1, but actually the shortest path from 0 to 1 is 0 -> 2 -> 3 -> 1 with the weight = 3 + 1 + (-8) = -4. Moreover, is we add 8 to all the edges in the graph, it will take to the different result.

Problem 3. *(20 points)* Extend the pseudocode of the Bellman-Ford algorithm given at the lecture so it can detect negative cycles.

```
def Bellman-Ford-modified (G,s,w):
        Init-SS (G,s)
        for i=1 to |G.V|-1
                        for each edge (u,v) in G.E
                                RELAX (u,v,w)
        for each edge(u, v) in E
              if(v.distance > u.distance + weight(u, v)):
                        Print " A negative weight cycle exists"
```

Problem 4. *(50 points)* Write a program implementing Dijksta's algorithm. Upload your source code. Show your input graph and the obtained shortest path spanning tree in the space below.

To compile and run the file, change the directory to the folder where you saved it and run the following command lines:

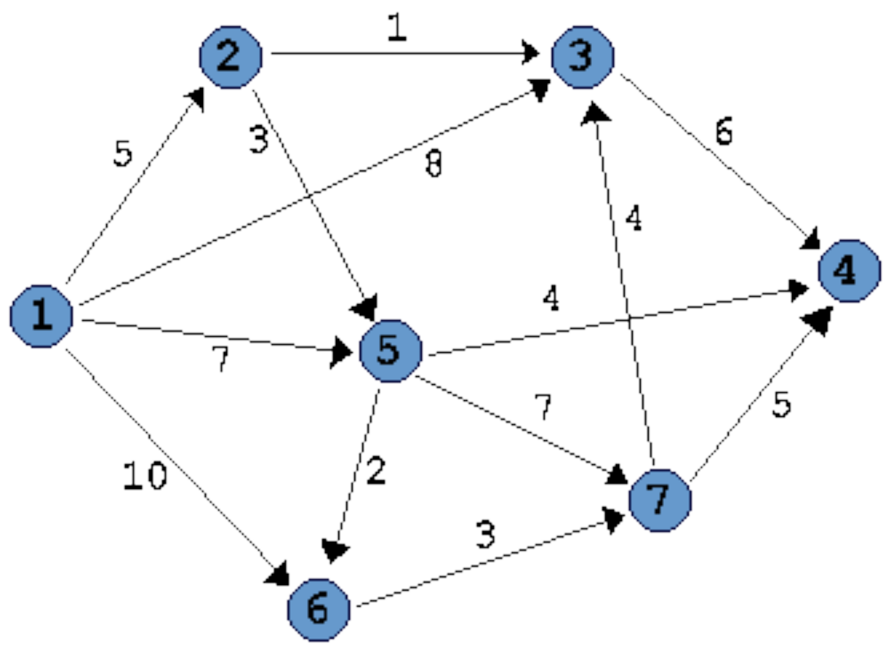
```
g++ -std=c++11 -o Dijkstra.o Dijkstra.cpp
./Dijkstra.o
```

Input: Firstly input the number of vertices in the graph and then itput repectively each cells of the weght matrix of the graph.

Output: each vertex with its own parent vertex and the shortest distance from source vertex to itself for the source vertex , its parent will be -1 to mark that it is the source one.

In the code, set 0 vertex to be the source vertex to start travelling, so change it if you want (be sure that it is smaller then the number of vertex)

For example, for the graph:



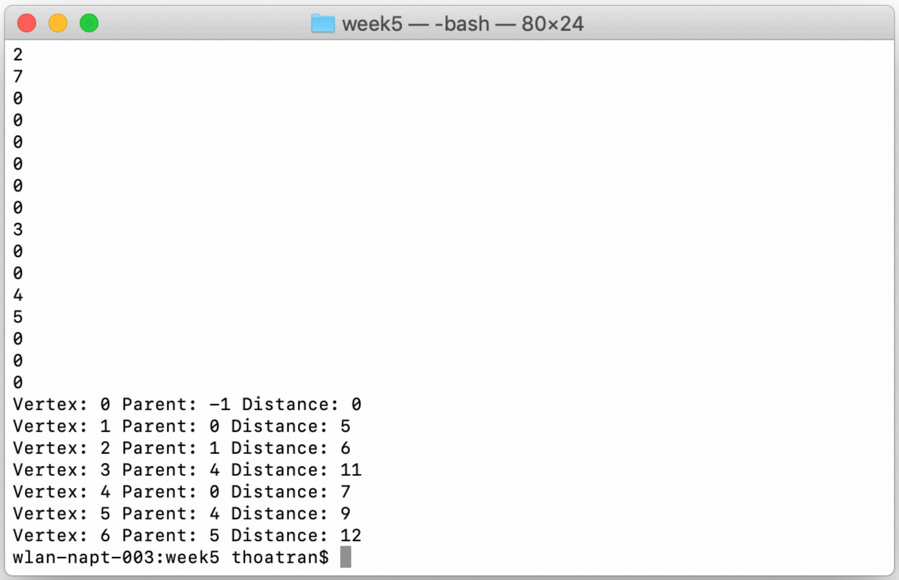
Graph D

The input will be:

```
7        //number of vertices
0 5 8 0 7 10 0         //the weight matrix
0 0 1 0 3 0 0
0 0 0 6 0 0 0
0 0 0 0 0 0 0
0 0 0 4 0 2 7
0 0 0 0 0 0 3
0 0 4 5 0 0 0
```

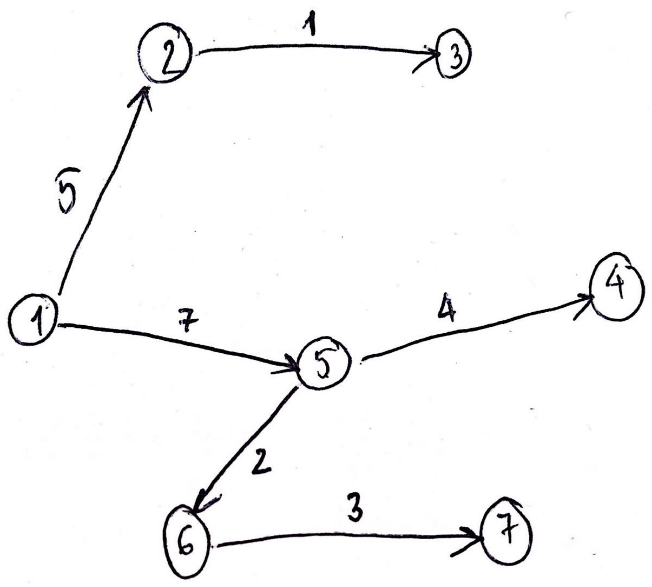
The output when staring traveling from vertex 1 is:

(note that the indexes in the code is from 0 but the indexes in the above graph is from 1 so in indexes in the output is equal to the indexes in the graph decreasing by 1)



```
2
7
0
0
0
0
0
0
3
0
0
4
5
0
0
0
Vertex: 0 Parent: -1 Distance: 0
Vertex: 1 Parent: 0 Distance: 5
Vertex: 2 Parent: 1 Distance: 6
Vertex: 3 Parent: 4 Distance: 11
Vertex: 4 Parent: 0 Distance: 7
Vertex: 5 Parent: 4 Distance: 9
Vertex: 6 Parent: 5 Distance: 12
wlan-napt-003:week5 thoatran$
```

So we have the shortest path spanning tree:


$$\begin{tikzcd}
 & 2 \arrow[rr, "1"] & & 3 & \\
1 \arrow[ur, "5"] \arrow[rr, "7"] & & 5 \arrow[rr, "4"] \arrow[dl, "2"] & & 4 \\
 & 6 \arrow[rr, "3"] & & 7 &
\end{tikzcd}$$