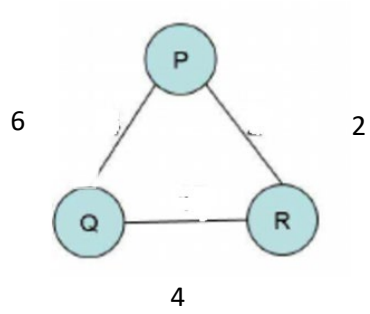
Task 6

Max Team Size 2

Team Members: _____

In this practice, we will implement the Bellman-Ford Shortest path algorithm.

i) First, we are going to use the adjacency list representation for graph. We are going to declare a dictionary G to represent the graph and the keys in the dictionary are going to be the nodes in the graph. The values associated with a key is another dictionary. In this second dictionary the key is the vertices that are adjacent to the vertex and the value is the weight of the edge connecting the vertices. For example, if we have the following graph the dictionary is going to be,
G = {'P': {'Q': 6, 'R': 2}, 'Q': {'P':6,'R':4}, 'R': {'P':2,'Q',4}}



Note:

1. If you want to get the weight of the PQ, you can use the syntax G['P']['Q']
2. You can assign infinity by using float('inf') to the right hand side of the assignment.

ii) We discussed about the Bellman-Ford algorithm in class. Write a python function called **Bellman_Ford** that takes graph and the starting vertex and returns the predV and distance dictionaries. You do not have to implement any class.

Function Header: **Bellman_Ford(Graph, startV)**

iii) Write a function that detects a negative weighted cycle in a graph also.

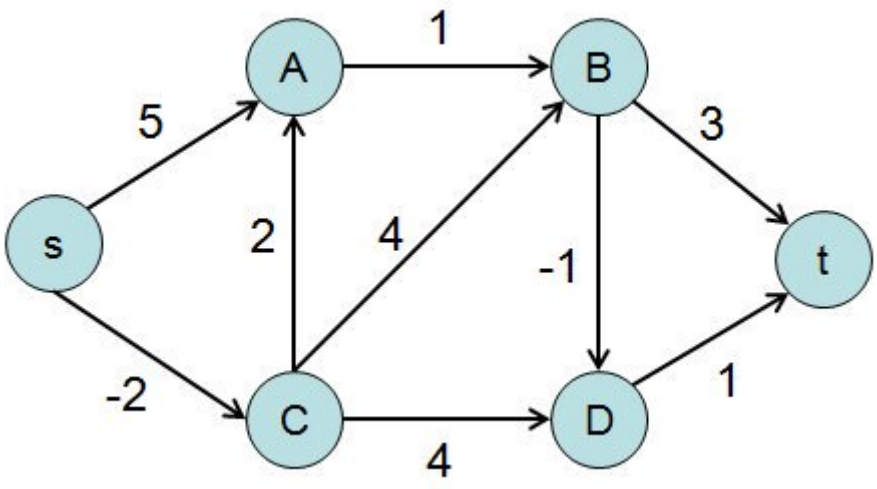
Function Header: **Detect_Negative_Cycle(Graph, startV)**

iv) Write a function **printallpaths** to print the path from a starting vertex to all other vertices in the graph. This function takes a starting vertex and the predV dictionary returned by the **Bellman_Ford** and then prints the paths to all other vertices in the following format. This function will not return any value.

For the following graph your paths should look like this.

s
s-->C-->A
s-->C-->A-->B
s-->C
s-->C-->A-->B-->D
s-->C-->A-->B-->D-->t

v) Write a main function, and create the following graph



vi) Call the **Bellman_Ford** function with starting vertex ‘s’.

Note:

There is no incoming edge to ‘s’ and for C the only incoming edge is from ‘s’, hence if you try to find the path from other vertices, you might not find a path (although Bellman_Ford will give you a path but not from the starting vertex)

vii) Call the Print the paths to all other vertices from s.