CSC-507 Programming Lab

Week 2

Instructions: Please read the following instructions **carefully** before attempting the questions.

1. Make a folder named 'RollNumber FirstName LastName' eg. '1101101 Alice Bob'
2. Save the solution code for problem X as 'probX.cpp'.
3. Make sure it runs correctly, using input from prompt in a file 'inp.txt' as 'a.out < inp.txt'

Problem 1: Graph Algorithms using Preprocessor directives. **[10 Marks]**
Write a single code which when compiled with the following flags perform as follows:

- $-D\ BFS$: Input Unweighted graph, Output BFS tree (using Queue) starting from 0.
- $-D\ DFS$: Input Unweighted graph, Output DFS tree (using Stacks) starting from 0.
- $-D\ DJK$: Input Weighted Graph, Output Shortest Path tree from 0 using Dijikstra's algorithm.
- $-D\ WBFS$: Input Weighted Graph with integer weight $0 - c$ (for some given constant c), Output shortest path tree starting from 0 using modified BFS algorithm (hint see 0/1 BFS).
- $-D\ DIR$: Applicable for above four cases, considers the input graph as directed.
- $-D\ PRM$: Input Weighted Undirected Graph, Output Minimum Spanning Tree using Prim's algorithm.

Give different definitions of Macros for the above using preprocessor directives $\#ifdef$ etc., which are then used in the same code uniformly across all algorithms.

Problem 2: Graph Algorithms using OOPs. **[10 Marks]**
Write a common class file to use the above algorithms, using extended (or inherited) classes to define only the differences. The choice of algorithm used and undirected/directed graph can be based on a flag passed during object creation.

Input	**BFS**	**DFS**
6 8	0 0	0 0
0 1	1 1	1 1
0 4	4 1	2 2
3 5	2 2	3 3
3 0	3 2	5 4
1 2	5 3	4 5
4 5		
2 3		
4 3		

(a) Unweighted Undirected

Input	**Dijkstra**	**Prim**
6 8	0 1	0 1
0 1 5	0 4	1 2
0 4 13	0 3	0 4
3 5 9	1 2	4 5
3 0 15	4 5	5 3
1 2 10		
4 5 8		
2 3 20		
4 3 25		

(b) Weighted Undirected

Input	**WBFS**
6 8	0 1
0 1 1	0 4
0 4 1	4 3
3 5 0	3 5
3 0 0	1 2
1 2 1	
4 5 1	
2 3 0	
4 3 0	

(c) Weighted Directed

Table 1: Input Output Sample

Graph. Define a $vector < list < pair < int, int >>> adj$, where an element (j, c) of $adj[i]$ describes an edge (i, j) with weight c. Elements of a pair $pair < int, int > p$ are accessed as $p.first$ and $p.second.$

DFS/BFS Input Output Format. Take as input the number of vertices n and edges m, followed by m lines, each describing an unweighted edge $u\ v$. Output each vertex in visit order with its level in a new line.

Dijkstra/WBFS/Prim Input Output Format. Take as input the number of vertices n and edges m, followed by m lines, each describing a weighted edge $u\ v\ c$, where $w(u, v) = c$. Output the edges of the tree in the order of addition each in a new line.