Practical 8

Aim:

Implement All pair shortest path

Code:

```
#include <iostream>
#include <vector>
#include <climits>
using namespace std;

#define INF INT_MAX

class Graph {
   int V;
   vector<vector<int>> dist;

public:
   Graph(int V) {
      this->V = V;
      dist.assign(V, vector<int>(V, INF));

      // Initializing distances to 0 for self-loops
      for (int i = 0; i < V; ++i)
         dist[i][i] = 0;
   }

   void addEdge(int u, int v, int weight) {
      dist[u][v] = weight;
      // For an undirected graph, uncomment the line below
      // dist[v][u] = weight;
   }

   void floydWarshall() {
      for (int k = 0; k < V; ++k) {
         for (int i = 0; i < V; ++i) {
            for (int j = 0; j < V; ++j) {
               if (dist[i][k] != INF && dist[k][j] != INF && dist[i][j] > dist[i][k] + dist[k][j]) {
                  dist[i][j] = dist[i][k] + dist[k][j];
               }
            }
         }
      }
   }

   void printShortestDistances() {
      cout << "Shortest distances between all pairs of vertices:\n";
```

Practical 8

```
        for (int i = 0; i < V; ++i) {
            for (int j = 0; j < V; ++j) {
                if (dist[i][j] == INF)
                    cout << "INF\t";
                else
                    cout << dist[i][j] << "\t";
            }
            cout << endl;
        }
    }
};

int main() {
    int V, E; // V -> Number of vertices, E -> Number of edges
    cout << "Enter the number of vertices and edges: ";
    cin >> V >> E;

    Graph g(V);

    cout << "Enter edges with weights (format: u v weight):" << endl;
    for (int i = 0; i < E; ++i) {
        int u, v, weight;
        cin >> u >> v >> weight;
        g.addEdge(u, v, weight);
    }

    g.floydWarshall();
    g.printShortestDistances();

    return 0;
}
```

Output: