lab 8 :

Aim : Implement Dijkstras algorithm to compute the shortest path for a given topology.

```
#include <iostream>
#include <stdio.h>
using namespace std;
#define V 4
int minDistance (int dist[], bool sptSet[])
{ int min = INT_MAX, min_index;
  for (int v=0; v < V; v++)
    if ( sptSet[v] == false && dist[v] <= min)
      min = dist[v], min_index = v;
  return min_index;
}
void dijkstra (int graph[V][V], int src)
{ int dist[V];
  bool sptSet[V];
  for (int i=0; i<V; i++)
    dist[i] = INT_MAX, sptSet[i] = false;
  dist[src]=0;
```

```
for (int count=0; count<V-1; count++){
    int u = minDistance(dist, sptSet);

    sptSet[u] = true;

    for (int v=0; v<V; v++)
        if (!sptSet[v] && graph[u][v] && dist[u]
                != INT_MAX && dist[u]+graph
                < dist[v])

            dist[v] = dist[u] + graph[u][v];
}

printSolution(dist);
}

int main(){

}

void printSolution(int dist[])
{
    cout << "Vertex \t\t Distance from Source\n";
    for (int i=0; i<V; i++)
        cout << i << "\t\t" << dist[i];
}
```

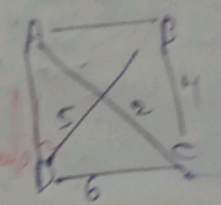
```
int main()
{
    int graph[V][V]
    cout << "Enter the graph" << endl;

    for (int i=0 ; i<V ; i++)
    {
        for (int j=0 ; j<V ; j++)
            cin >> graph[i][j];
    }
    dijkstra(graph, 0);
    return 0;
}
```

Output :

Enter the graph

0	1	2	3
1	0	4	5
2	4	0	6
3	5	6	0



Vertex	Distance from source
0	0
1	1
2	2
3	3