Sumalata
1BM19CS409

Date ___/___/___

10) Implement Dijkstra's algorithm to compute the shortest path through a graph.

```
#include < iostream >
#include < climits >
using namespace std;
int a[30) (30), n;
int minimum (int visited [], int dist())
{
    int mindist=10000, mini;
    for (int i=0; i<n; i++)
    {
        if (!visited (i) && dist (i) <mindis)
        {
            mindis = dist (i);
            mini = i;
        }
    }
return mini;
}
 void dijkstra (int src)
 {
    int dist(n), visited (n);
    for (int i=0; i<n; i++)
    {
```

Page No.

Date ___/___/___

```
    dist[i] = 10000;
    visited [i] = 0;
  }
  dist [src] = 0;
  for (int i=0; i<n; i++)
  {
      int u=minimum (visited, dist);
      visited [u] = 1;
      for (int v=0; v<n; v++)
      {
      if (!visited[v] && a[u][v] != 10000
      && dist[u] != 10000 && (dist[u] + a[u][v])
          < dist[v])
      dist[v] = dist[u] + a[u][v];
      }
  }
  cout << "Shortest Paths to all other vertices from
      " << src << " is " << endl;
  cout << "vertices \t Distance from Source" << endl;
  for (int i=0; i<n; i++)
  {
      if (i!=src)
          cout << i << "\t\t" << dist[i] << endl;
  }
}
```

Page No.

Date ___/___/______

```
int main()
{
cout << "enter the no. of vertices " <<endl;
cin >> n;
cout <<" enter the weighted adjancency matrix
 (enter 10000 if there is no edge) " <<endl;
 for (int i=0; i<n; i++)
  {
     for (int j=0; j<n; j++)
       cin >> a[i][j];
  }
 int src;
 cout <<" enter the source vertex" <<endl;
 cin >> src;
 dijkstra (src);
 return 0;
}
```

Page No.