**LAB-6**

**From a given vertex in a weighted connected graph, find shortest paths to other  
vertices using Dijkstra’s algorithm**.

#include <stdio.h>

int cost[10][10], n, result[10][2], weight[10];

void dijkstras(int [][10], int );

int main()

{

int i, j, s;

printf("Enter the number of vertices: ");

scanf("%d", &n);

printf("Enter the cost adjacency matrix:\n");

for (i = 0; i < n; i++)

{

for (j = 0; j < n; j++)

{

scanf("%d", &cost[i][j]);

}

}

printf("Enter the source vertex: ");

scanf("%d", &s);

dijkstras(cost, s);

printf("Path:\n");

for (i = 1; i < n; i++) {

printf("(%d, %d) with weight %d ", result[i][0], result[i][1], weight[result[i][1]]);

}

return 0;

}

void dijkstras(int cost[][10], int s)

{

int d[10], p[10], visited[10];

int i, j, min, u, v, k;

for(i = 0; i < 10; i++)

{

d[i] = 999;

visited[i] = 0;

p[i] = s;

}

d[s] = 0;

visited[s] = 1;

for(i = 0; i < n; i++)

{

min = 999;

u = 0;

for(j = 0; j < n; j++)

{

if(visited[j] == 0)

{

if(d[j] < min)

{

min = d[j];

u = j;

}

}

}

visited[u] = 1;

for(v = 0; v < n; v++)

{

if(visited[v] == 0 && (d[u] + cost[u][v] < d[v]))

{

d[v] = d[u] + cost[u][v];

p[v] = u;

}

}

}

for(i = 0; i < n; i++)

{

result[i][0] = p[i];

result[i][1] = i;

weight[i] = d[i];

}

}

**OUTPUT:**

