S.No: 20	Exp. Name: ***Shortest Path from source to vertex***	Date:2025-10-23

Aim:

You are required to write a C program that computes the shortest paths from a source vertex to all other vertices in a weighted, directed graph. The program should utilize Dijkstra's algorithm to solve this problem.

The program must perform the following tasks:
Input the Number of Vertices:

- Print: "Enter the number of vertices: "
- Read an integer V representing the number of vertices in the graph.

Input the Adjacency Matrix:

- Print: **"**adjacency matrix:**"**
- For each pair of vertices (i, j), print: **"**Weight from i to j: **"**
- Read the weight of the edge between vertex i and vertex j. If there is no edge between them, input 0.

Input the Source Vertex:

- Print: **"**Enter the source vertex: **"**
- Read the source vertex src.

Validation:

- Ensure that the number of vertices does not exceed 100. If it does, print: "Number of vertices exceeds maximum allowed (100)" and exit the program.
- Ensure that the source vertex is within the valid range (0 to V-1). If it is not, print: "Invalid source vertex" and exit the program.

Compute Shortest Paths:

- Use Dijkstra's algorithm to compute the shortest paths from the source vertex to all other vertices.

Note: Refer to sample test cases for better understanding

Source Code:

Path.c

```
#include <stdio.h>
#include <limits.h>
#include <stdbool.h>

#define MAX_V 100

int minDistance(int dist[], bool sptSet[], int V) {

   // Write your code here...
   int min = INT_MAX, min_index = -1;
   for(int v = 0; v<V; v++){
      if(!sptSet[v] && dist[v] <= min){
         min = dist[v];
         min_index = v;
      }
   }
```

```
   return min_index;

}

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

void dijkstra(int graph[MAX_V][MAX_V], int src, int V) {

   // Write your code here...
   int dist[MAX_V];
   bool sptSet[MAX_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, V);
      if(u == -1){
         break;
      }
      sptSet[u] = true;
      for(int v = 0; v<V; v++){
         if(!sptSet[v] && graph[u][v] > 0 && dist[u]!= INT_MAX && dist[u] + graph[u]
[v] < dist[v]){
            dist[v] = dist[u] + graph[u][v];
         }
      }
   }

   printSolution(dist,V);


}

int main() {
    int graph[MAX_V][MAX_V];
    int V, src;

    printf("number of vertices: ");
    scanf("%d", &V);

    if (V > MAX_V) {
        printf("Number of vertices exceeds maximum allowed (%d)\n", MAX_V);
        return 1;
    }

    for (int i = 0; i < V; i++) {
```

```
        for (int j = 0; j < V; j++) {
            graph[i][j] = 0;
        }
    }

    printf("adjacency matrix:\n");
    for (int i = 0; i < V; i++) {
        for (int j = 0; j < V; j++) {
            printf("Weight from %d to %d: ", i, j);
            scanf("%d", &graph[i][j]);
        }
    }

    printf("Enter the source vertex: ");
    scanf("%d", &src);

    if (src < 0 || src >= V) {
        printf("Invalid source vertex\n");
        return 1;
    }

    dijkstra(graph, src, V);

    return 0;
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
number of vertices: 3
adjacency matrix: 0
Weight from 0 to 0: 0
Weight from 0 to 1: 5
Weight from 0 to 2: 10
Weight from 1 to 0: 5
Weight from 1 to 1: 0
Weight from 1 to 2: 2
Weight from 2 to 0: 10
Weight from 2 to 1: 2
Weight from 2 to 2: 0
Enter the source vertex: 0
Vertex Distance from Source
0 0
1 5
2 7