```
#include <iostream>
#define INF 9999
using namespace std;

void dijkstra(int graph[10][10], int n, int start) {
  int distance[10], visited[10];

  // Initialization
  for (int i = 0; i < n; i++) {
    distance[i] = INF;
    visited[i] = 0;
  }

  distance[start] = 0;

  for (int count = 0; count < n - 1; count++) {
    // Find the minimum distance unvisited vertex
    int min = INF, u;

    for (int i = 0; i < n; i++) {
      if (!visited[i] && distance[i] <= min) {
        min = distance[i];
        u = i;
      }
    }
```

```
        visited[u] = 1;

        // Update distance of adjacent vertices
        for (int v = 0; v < n; v++) {
            if (!visited[v] && graph[u][v] && distance[u] != INF
                && distance[u] + graph[u][v] < distance[v]) {
                distance[v] = distance[u] + graph[u][v];
            }
        }
    }

    // Print shortest distances
    cout << "Vertex\tDistance from Source\n";
    for (int i = 0; i < n; i++)
        cout << i << "\t" << distance[i] << endl;
}

// -------- Main Function --------
int main() {
    int n;
    cout << "Enter number of vertices: ";
    cin >> n;

    int graph[10][10];
    cout << "Enter adjacency matrix (0 for no edge):\n";
    for (int i = 0; i < n; i++)
```

```
        for (int j = 0; j < n; j++)
            cin >> graph[i][j];

    int start;
    cout << "Enter source vertex (0 to " << n-1 << "): ";
    cin >> start;

    dijkstra(graph, n, start);

    return 0;
}
```