Name	DARSHIT BHAGTANI
UID	2021700006
Experiment No.	6

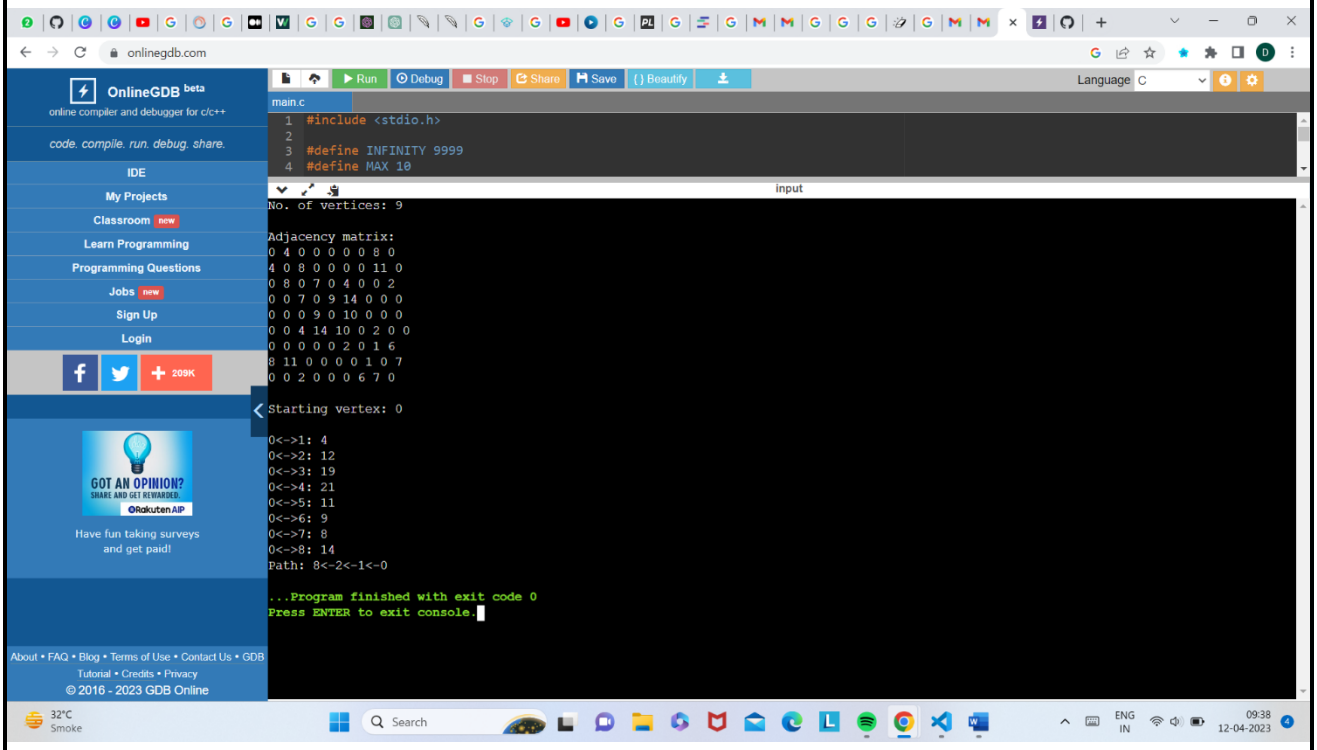
AIM:	Greedy Approach- Single Source Shortest path-Dijkstra's Algorithm
PROBLEM STATEMENT :	Find the shortest path from a single source vertex to all other vertices in a weighted graph using Dijkstra's algorithm
ALGORITHM/ THEORY:	**Dijkstra's algorithm is a popular algorithm for solving the single-source shortest path problem in weighted graphs with non-negative edge weights.** **The algorithm starts at a source vertex and iteratively explores the vertices that are closest to the source vertex until all vertices have been explored.** **The algorithm may or may not work for negative edge weights.** **Algorithm:** **1. Initialize all vertices with a distance of infinity except the source vertex, which is assigned a distance of 0.** **2. Repeat the following n times:** **a. Select the unvisited vertex with the smallest tentative distance.** **b. Mark the selected vertex as visited.** **c. For each neighbor of the selected vertex that is not yet visited, calculate the tentative distance to that neighbor by adding the distance between the selected vertex and the neighbor to the selected vertex's tentative distance.** **d. If the tentative distance is smaller than the current distance of the neighbor, update the neighbor's distance.** **3. Return the distances from the source vertex to all other vertices.**

PROGRAM:	

```c
#include <stdio.h>

#define INFINITY 9999
#define MAX 10

void dijkstra(int graph[MAX][MAX], int n, int sV)
{
    int cost[MAX][MAX], distance[MAX], pred[MAX];
    int visited[MAX], count, mDis, nV, i, j;
    int t = 0;
    for (i = 0; i < n; i++)
        for (j = 0; j < n; j++)
            if (graph[i][j] == 0)
                cost[i][j] = INFINITY;
            else
                cost[i][j] = graph[i][j];
    for (i = 0; i < n; i++)
    {
        distance[i] = cost[sV][i];
        pred[i] = sV;
        visited[i] = 0;
    }
    distance[sV] = 0;
    visited[sV] = 1;
    count = 1;
    while (count < n - 1)
    {
        mDis = INFINITY;
        for (i = 0; i < n; i++)
            if (distance[i] < mDis && !visited[i])
            {
                mDis = distance[i];
                nV = i;
            }
        visited[nV] = 1;
        for (i = 0; i < n; i++)
            if (!visited[i])
                if (mDis + cost[nV][i] < distance[i])
                {
                    distance[i] = mDis + cost[nV][i];
                    pred[i] = nV;
                }
        count++;
    }

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

```
        if (i != sV)
        {
            printf("\n%d<->%d: %d", sV, i, distance[i]);
            if (i == n - 1)
            {
                printf("\nPath: %d", i);
                j = i;
                while (j != sV)
                {
                    j = pred[j];
                    printf("<-%d", j);
                }
            }
        }
    }
}

int main()
{
    int graph[MAX][MAX], i, j, n, u;
    printf("No. of vertices: ");
    scanf("%d", &n);
    printf("\nAdjacency matrix: \n");
    for (i = 0; i < n; i++)
        for (j = 0; j < n; j++)
            scanf("%d", &graph[i][j]);
    printf("\nStarting vertex: ");
    scanf("%d", &u);
    dijkstra(graph, n, u);

    return 0;
}
```

RESULT:



CONCLUSION:	Successfully understood and implemented Dijkstra's algorithm in C.