```
#include<stdio.h>
#define SIZE 10
#define INFINITY 999


void read_graph(int *nv, int adj[][SIZE])
{
    int i, j;

    printf("\nEnter the number of vertices : ");
    scanf("%d", nv);

    printf("\nEnter the adjecency matrix (order %d x %d) :\n", *nv, *nv);
    for( i = 0; i < *nv; i++ )
        for( j = 0; j < *nv; j++ )
            scanf("%d", &adj[i][j]);
}


void Dijkstra(int adj[][SIZE], int *nv, int start, int distance[])
{
    int cost[SIZE][SIZE], pred[SIZE];
    int visited[SIZE], count, mindistance, nextnode, i, j;

    if( !*nv )
        {
            printf("\nPlease read a graph...\n");
            return;
        }

    for(i = 0; i < *nv; i++)
        for(j = 0; j < *nv; j++)
            if(adj[i][j] == 0)
                cost[i][j] = INFINITY;
            else
                cost[i][j] = adj[i][j];

    for(i = 0; i < *nv; i++)
    {
        distance[i] = cost[start][i];
        pred[i] = start;
        visited[i] = 0;
    }

    distance[start] = 0;
    visited[start] = 1;
    count = 1;

    while(count < *nv - 1)
    {
```

```
        mindistance = INFINITY;

        for(i = 0; i < *nv; i++)
            if(distance[i] < mindistance && !visited[i])
            {
                mindistance = distance[i];
                nextnode = i;
            }

        visited[nextnode] = 1;
        for(i = 0; i < *nv; i++)
            if(!visited[i])
                if(mindistance + cost[nextnode][i] < distance[i])
                {
                    distance[i] = mindistance + cost[nextnode][i];
                    pred[i] = nextnode;
                }

        count++;
    }

    printf("\nSuccessfully created shortest path vector beased on the given start
vertex %d \n", start);


    for(i = 0; i < *nv; i++)
        if(i != start)
        {
            printf("\nDistance from source to %d: %d", i, distance[i]);
        }

}

void display(int adj[][SIZE], int *nv, int flag , int distance[], int start)
{
    int i, j;

    if( !*nv )
        {
            printf("\nPlease read a graph...\n");
            return;
        }
    printf("\nThe given graph (adjacency matrix) is:\n");
        for( i = 0; i < *nv; i++ )
        {
                for( j = 0; j < *nv; j++ )
                        printf("%d ", adj[i][j] );
                printf("\n");
        }
```

```
        if(flag)
        {
            for(i = 0; i < *nv; i++)
        if(i != start)
        {
            printf("\nDistance from source to %d: %d", i, distance[i]);
        }
        }
}


int main()
{
        int adj[SIZE][SIZE], distance[SIZE];
        int nv;
        int start = 0;
        int flag = 0;
        int e = 1, ch;

        while( e )
        {
                printf( "\n--------------MENU--------------\n" );
                printf( "\n\t1. Read Graph\n\t2. Display\n\t3. Dijksta's Algorithm
- Shortest path(Single source)\n\t4. Exit\n" );
                printf( "\n-------------------------------\n" );
                printf( "\n Enter your choice:" );
                scanf( "%d", &ch );

                switch( ch )
                {
                        case 1: read_graph(&nv, adj);
                             break;
                        case 2: display(adj, &nv, flag, distance, start);
                                 break;
                        case 3: flag = 1;
                                Dijkstra(adj, &nv, start, distance);
                             break;
                        case 4 : e = 0;
                                 break;
                        default: printf( "\n Invalid choice \n" );
                }

        }
        return 0;
}
```