Programmer Name: Param Chordiya

Batch: E5

Problem Statement:

Write a program in C to represent graph using adjacency list or matrix and generate minimum spanning tree using Prim's algorithm.

INPUT:

```
#include<stdio.h>
#include<stdlib.h>

#define infinity 9999
#define MAX 20

int G[MAX][MAX],spanning[MAX][MAX],n;

int prims();

void main()
{
                printf("\n***************************************");
                printf("\n        ROLL NO:22119\n");
                printf("***************************************\n");
                int i,j,total_cost;
                printf("Enter no. of vertices:");
                scanf("%d",&n);

                printf("\nEnter the adjacency matrix:\n");

                for(i=0;i<n;i++)
                    for(j=0;j<n;j++)
                            scanf("%d",&G[i][j]);

                total_cost=prims();
                printf("\nMinimum spanning tree matrix:\n");

                for(i=0;i<n;i++)
                {
                    printf("\n");
                    for(j=0;j<n;j++)
                            printf("%d\t",spanning[i][j]);
                }
```

```
        printf("\n\nMinimum Spanning tree=%d",total_cost);

}

int prims()
{
        int cost[MAX][MAX];
        int u,v,min_distance,distance[MAX],from[MAX];
        int visited[MAX],no_of_edges,i,min_cost,j;


        for(i=0;i<n;i++)
            for(j=0;j<n;j++)
            {
                    if(G[i][j]==0)
                            cost[i][j]=infinity;
                    else
                            cost[i][j]=G[i][j];
                            spanning[i][j]=0;
            }


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

        for(i=1;i<n;i++)
        {
            distance[i]=cost[0][i];
            from[i]=0;
            visited[i]=0;
        }

        min_cost=0;
        no_of_edges=n-1;

        while(no_of_edges>0)
        {

            min_distance=infinity;
            for(i=1;i<n;i++)
                    if(visited[i]==0&&distance[i]<min_distance)
                    {
                            v=i;
```

```
                    min_distance=distance[i];
                }

            u=from[v];

            spanning[u][v]=distance[v];
            spanning[v][u]=distance[v];
            no_of_edges--;
            visited[v]=1;


            for(i=1;i<n;i++)
                if(visited[i]==0&&cost[i][v]<distance[i])
                {
                    distance[i]=cost[i][v];
                    from[i]=v;
                }

            min_cost=min_cost+cost[u][v];
        }

        return(min_cost);
}
```

OUTPUT:

```
************************************
          ROLL NO:22119
************************************
Enter no. of vertices:6

Enter the adjacency matrix:
0 3 1 6 0 0
3 0 5 0 3 0
1 5 0 5 6 4
6 0 5 0 0 2
0 3 6 0 0 6
0 0 4 2 6 0

Minimum spanning tree matrix:

0       3       1       0       0       0
3       0       0       0       3       0
1       0       0       0       0       4
0       0       0       0       0       2
0       3       0       0       0       0
0       0       4       2       0       0

Minimum Spanning tree=13
--------------------------------
Process exited after 50.95 seconds with return value 26
Press any key to continue . . .
```