# Experiment-4

```
// A C program for Prim's Minimum
// Spanning Tree (MST) algorithm. The program is
// for adjacency matrix representation of the graph

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

// Number of vertices in the graph
#define V 5

int minKey(int key[], bool mstSet[])
{
    // Initialize min value
    int min = INT_MAX, min_index;

    for (int v = 0; v < V; v++)
        if (mstSet[v] == false && key[v] < min)
            min = key[v], min_index = v;

    return min_index;
}

int printMST(int parent[], int graph[V][V])
{
    printf("Edge \tWeight\n");
    for (int i = 1; i < V; i++)
        printf("%d - %d \t%d \n", parent[i], i,
               graph[i][parent[i]]);
}

// Function to construct and print MST for
// a graph represented using adjacency
// matrix representation
void primMST(int graph[V][V])
{
    // Array to store constructed MST
    int parent[V];
    int key[V];
```

```
    // To represent set of vertices included in MST
    bool mstSet[V];

    // Initialize all keys as INFINITE
    for (int i = 0; i < V; i++)
        key[i] = INT_MAX, mstSet[i] = false;

    key[0] = 0;

    parent[0] = -1;

    // The MST will have V vertices
    for (int count = 0; count < V - 1; count++) {

        int u = minKey(key, mstSet);

        // Add the picked vertex to the MST Set
        mstSet[u] = true;

        for (int v = 0; v < V; v++)

            if (graph[u][v] && mstSet[v] == false
                && graph[u][v] < key[v])
                parent[v] = u, key[v] = graph[u][v];
    }

    // print the constructed MST
    printMST(parent, graph);
}

// Driver's code
int main()
{
    int graph[V][V] = { { 0, 2, 0, 6, 0 },
                        { 2, 0, 3, 8, 5 },
                        { 0, 3, 0, 0, 7 },
                        { 6, 8, 0, 0, 9 },
                        { 0, 5, 7, 9, 0 } };

    primMST(graph);

    return 0;
}
```

# Experiment-5

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

const int inf = INT_MAX;
int k, a, b, u, v, n, ne = 1;
int mincost = 0;
int cost[5][5] = {{0,1,0,10,0},
                  {0,0,3,0,0},
                  {7,0,0,4,0},
                  {0,0,0,0,2},
                  {5,0,0,0,0}};
int  p[5] = {0};
int applyfind(int i)
{
    while(p[i] != 0)
        i=p[i];
    return i;
}
int applyunion(int i,int j)
{
    if(i!=j) {
        p[j]=i;
        return 1;
    }
    return 0;
```

```
}
int main(void)
{
    n = 5;
    int i, j;
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            if (cost[i][j] == 0) {
                cost[i][j] = inf;
            }
        }
    }
    printf("Minimum Cost Spanning Tree:
\n");
    while(ne < n) {
        int min_val = inf;
        for(i=0; i<n; i++) {
            for(j=0; j <n; j++) {
                if(cost[i][j] <
min_val) {
                    min_val = cost[i]
[j];
                    a = u = i;
                    b = v = j;
                }
            }
        }
        u = applyfind(u);
        v = applyfind(v);
        if(applyunion(u, v) != 0) {
```

```
            printf("%d -> %d   %d\n",
a, b,min_val);
            mincost +=min_val;
        }
        cost[a][b] = cost[b][a] = inf;
        ne++;
    }
    printf("Minimum cost =
%d\n",mincost);
    return 0;
}
```