DATA STRUCTURES

Division	**CS(AIML) -A**
Batch	**2**
GR-no	**12311493**
Roll no	**54**
Name	**Atharva Kangralkar**

Assignment 12:

Generate Minimum Spanning Tree Using Prims Algorithm when Graph is Represented using A. Adjacency Matrix. B. Adjacency Lists.

Code:-

1. Adjacency Matrix

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

#define MAX 100

int adjMatrix[MAX][MAX];
int parent[MAX];
int key[MAX];
int visited[MAX];

void primMST(int n) {
  for (int i = 0; i < n; i++) {
    key[i] = INT_MAX;
    visited[i] = 0;
    parent[i] = -1;
  }
  key[0] = 0;

  for (int count = 0; count < n - 1; count++) {
    int u = -1;
    for (int i = 0; i < n; i++) {
      if (!visited[i] && (u == -1 || key[i] < key[u])) {
        u = i;
      }
    }
    visited[u] = 1;

    for (int v = 0; v < n; v++) {
      if (adjMatrix[u][v] != 0 && !visited[v] && adjMatrix[u][v] < key[v]) {
        key[v] = adjMatrix[u][v];
        parent[v] = u;
      }
    }
  }
```

```
    printf("Minimum Spanning Tree (Adjacency Matrix):\n");
    for (int i = 1; i < n; i++) {
        printf("%d - %d: %d\n", parent[i], i, adjMatrix[i][parent[i]]);
    }
}

int main() {
    int n, e, u, v, w;
    printf("Enter the number of vertices: ");
    scanf("%d", &n);
    printf("Enter the number of edges: ");
    scanf("%d", &e);

    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            adjMatrix[i][j] = 0;
        }
    }

    printf("Enter the edges (u v w) for an undirected graph:\n");
    for (int i = 0; i < e; i++) {
        scanf("%d %d %d", &u, &v, &w);
        adjMatrix[u][v] = w;
        adjMatrix[v][u] = w;
    }

    primMST(n);

    return 0;
}
```

2. Adjacency List

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

#define MAX 100

typedef struct Node {
```

```
    int vertex;
    int weight;
    struct Node* next;
} Node;

Node* adjList[MAX];
int parent[MAX];
int key[MAX];
int visited[MAX];

Node* createNode(int v, int weight) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    newNode->vertex = v;
    newNode->weight = weight;
    newNode->next = NULL;
    return newNode;
}

void addEdge(int u, int v, int w) {
    Node* newNode = createNode(v, w);
    newNode->next = adjList[u];
    adjList[u] = newNode;

    newNode = createNode(u, w);
    newNode->next = adjList[v];
    adjList[v] = newNode;
}

void primMST(int n) {
    for (int i = 0; i < n; i++) {
        key[i] = INT_MAX;
        visited[i] = 0;
        parent[i] = -1;
    }
    key[0] = 0;

    for (int count = 0; count < n - 1; count++) {
        int u = -1;
        for (int i = 0; i < n; i++) {
            if (!visited[i] && (u == -1 || key[i] < key[u])) {
                u = i;
```

```
        }
      }
      visited[u] = 1;

      Node* temp = adjList[u];
      while (temp != NULL) {
        int v = temp->vertex;
        if (!visited[v] && temp->weight < key[v]) {
          key[v] = temp->weight;
          parent[v] = u;
        }
        temp = temp->next;
      }
    }

    printf("Minimum Spanning Tree (Adjacency List):\n");
    for (int i = 1; i < n; i++) {
      printf("%d - %d: %d\n", parent[i], i, key[i]);
    }
}

int main() {
    int n, e, u, v, w;
    printf("Enter the number of vertices: ");
    scanf("%d", &n);
    printf("Enter the number of edges: ");
    scanf("%d", &e);

    for (int i = 0; i < n; i++) {
      adjList[i] = NULL;
    }

    printf("Enter the edges (u v w) for an undirected graph:\n");
    for (int i = 0; i < e; i++) {
      scanf("%d %d %d", &u, &v, &w);
      addEdge(u, v, w);
    }

    primMST(n);

    return 0;
```

```
}
```

Output:-

1. **Adjacency matrix:-**

```
Enter the number of vertices: 3
Enter the number of edges: 3
Enter the edges (u v w) for an undirected graph:
0 1 1
0 2 2
1 2 2
Minimum Spanning Tree (Adjacency Matrix):
0 - 1: 1
0 - 2: 2
```

2. **Adjacency list:-**

```
Enter the number of vertices: 4
Enter the number of edges: 4
Enter the edges (u v w) for an undirected graph:
0 1 5
1 2 6
2 3 7
3 0 8
Minimum Spanning Tree (Adjacency List):
0 - 1: 5
1 - 2: 6
2 - 3: 7
```