**Write a C program to create a graph and find a minimum spanning tree using prims algorithm.**

**Algorithm:**

**CODE:**

**#include <stdio.h>**

**#include <stdlib.h>**

**#include <limits.h>**

**#include <stdbool.h>**

**#define V 5**

**struct Edge {**

**int src, dest, weight;**

**};**

**struct Graph {**

**int V, E;**

**struct Edge\* edge;**

**};**

**struct Graph\* createGraph(int V, int E) {**

**struct Graph\* graph = (struct Graph\*) malloc(sizeof(struct Graph));**

**graph->V = V;**

**graph->E = E;**

**graph->edge = (struct Edge\*) malloc(graph->E \* sizeof(struct Edge));**

**return graph;**

**}**

**int minKey(int key[], bool mstSet[]) {**

**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;**

**}**

**void printMST(int parent[], struct Graph\* graph) {**

**printf("Edge \tWeight\n");**

**for (int i = 1; i < graph->V; i++)**

**printf("%d - %d \t%d \n", parent[i], i, graph->edge[i].weight);**

**}**

**void primMST(struct Graph\* graph) {**

**int parent[V];**

**int key[V];**

**bool mstSet[V];**

**for (int i = 0; i < V; i++)**

**key[i] = INT\_MAX, mstSet[i] = false;**

**key[0] = 0;**

**parent[0] = -1;**

**for (int count = 0; count < V - 1; count++) {**

**int u = minKey(key, mstSet);**

**mstSet[u] = true;**

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

**if (graph->edge[u].weight && mstSet[v] == false && graph->edge[u].weight < key[v])**

**parent[v] = u, key[v] = graph->edge[u].weight;**

**}**

**printMST(parent, graph);**

**}**

**int main() {**

**int V = 5;**

**int E = 7;**

**struct Graph\* graph = createGraph(V, E);**

**graph->edge[0].src = 0; graph->edge[0].dest = 1; graph->edge[0].weight = 2;**

**graph->edge[1].src = 0; graph->edge[1].dest = 3; graph->edge[1].weight = 6;**

**graph->edge[2].src = 1; graph->edge[2].dest = 2; graph->edge[2].weight = 3;**

**graph->edge[3].src = 1; graph->edge[3].dest = 3; graph->edge[3].weight = 8;**

**graph->edge[4].src = 1; graph->edge[4].dest = 4; graph->edge[4].weight = 5;**

**graph->edge[5].src = 2; graph->edge[5].dest = 4; graph->edge[5].weight = 7;**

**graph->edge[6].src = 3; graph->edge[6].dest = 4; graph->edge[6].weight = 9;**

**primMST(graph);**

**return 0;**

**}**

OUTPUT:

EDGE WEIGHT

* 1. 2
  2. 3
  3. 6

1-4 5