**Assignment 8**

**Name**: Komal Potdar

**Roll No**: 92

**PRN No**: 12320165

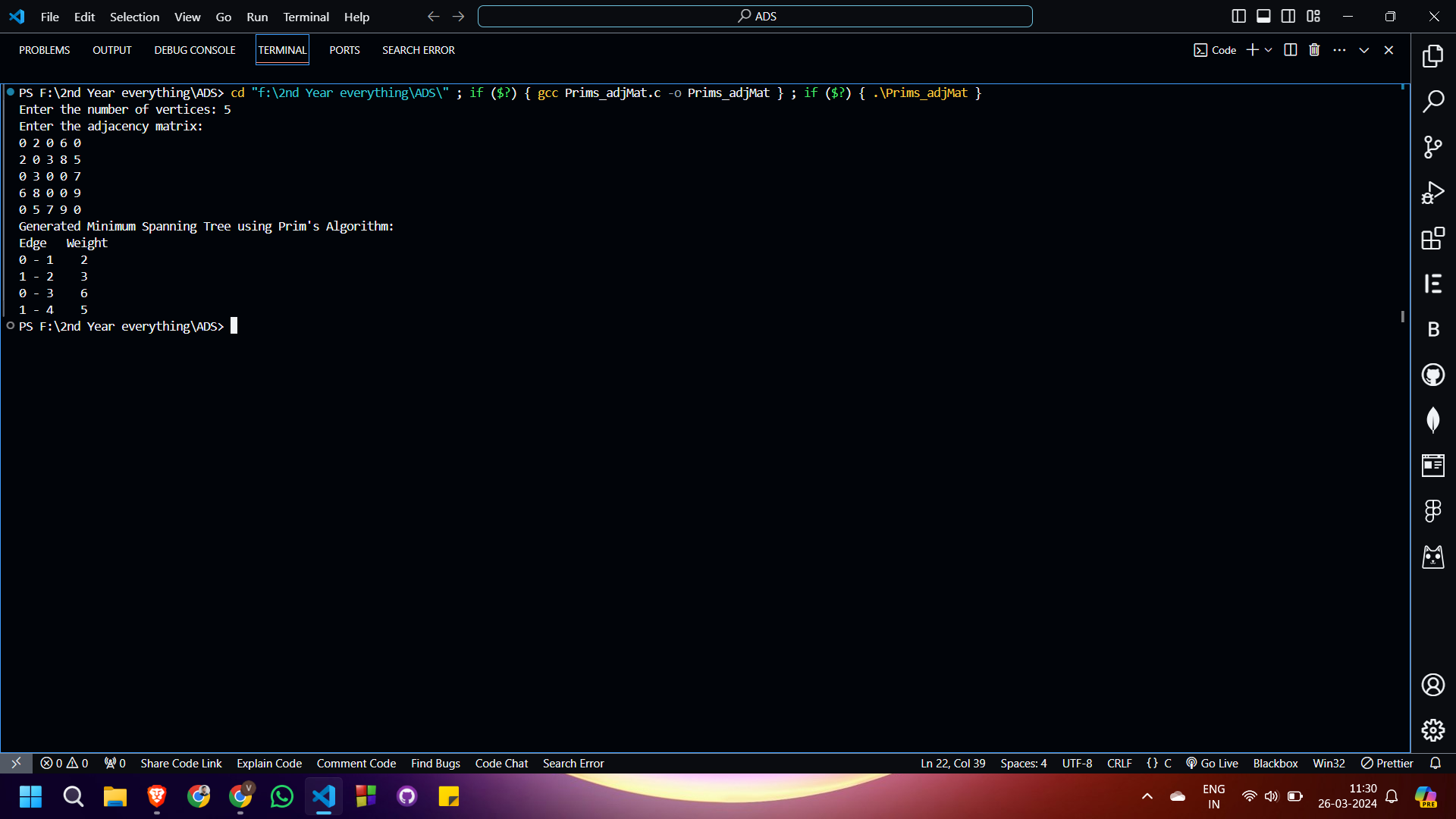
**Div**: CS B SY

**Batch**: 3

1. **Write a menu driven program in C to implement Prims Algorithm to generate MST using Adjacency Matrix.**

|  |
| --- |
| #include <stdio.h>  #include <stdbool.h>  #define MAX\_VERTICES 10  #define INF 999999  int minKey(int key[], bool mstSet[], int vertices) {      int min = INF, min\_index = -1; // Initialize min\_index to -1      for (int v = 0; v < vertices; v++) {          if (mstSet[v] == false && key[v] < min) {              min = key[v];              min\_index = v;          }      }      return min\_index;  }  void printMST(int parent[], int graph[MAX\_VERTICES][MAX\_VERTICES], int vertices) {      printf("Edge   Weight\n");      for (int i = 1; i < vertices; i++)          printf("%d - %d    %d \n", parent[i], i, graph[i][parent[i]]);  }  void primMST(int graph[MAX\_VERTICES][MAX\_VERTICES], int vertices) {      int parent[vertices];      int key[vertices];      bool mstSet[vertices];      for (int i = 0; i < vertices; i++) {          key[i] = INF;          mstSet[i] = false;      }      key[0] = 0;      parent[0] = -1;      for (int count = 0; count < vertices - 1; count++) {          int u = minKey(key, mstSet, vertices);          mstSet[u] = true;          for (int v = 0; v < vertices; v++) {              if (graph[u][v] && mstSet[v] == false && graph[u][v] < key[v]) {                  parent[v] = u;                  key[v] = graph[u][v];              }          }      }      printMST(parent, graph, vertices);  }  int main() {      int graph[MAX\_VERTICES][MAX\_VERTICES];      int vertices;      printf("Enter the number of vertices: ");      scanf("%d", &vertices);      printf("Enter the adjacency matrix:\n");      for (int i = 0; i < vertices; i++) {          for (int j = 0; j < vertices; j++) {              scanf("%d", &graph[i][j]);          }      }      printf("Generated Minimum Spanning Tree using Prim's Algorithm:\n");      primMST(graph, vertices);      return 0;  } |

Output:



1. **Implement Prims Algorithm to generate MST using Adjacency Lists.**

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  #include <limits.h>  #include <stdbool.h>  struct AdjListNode {      int dest;      int weight;      struct AdjListNode\* next;  };  struct AdjList {      struct AdjListNode\* head;  };  struct Graph {      int V;      struct AdjList\* array;  };  struct AdjListNode\* newAdjListNode(int dest, int weight) {      struct AdjListNode\* newNode = (struct AdjListNode\*)malloc(sizeof(struct AdjListNode));      newNode->dest = dest;      newNode->weight = weight;      newNode->next = NULL;      return newNode;  }  struct Graph\* createGraph(int V) {      struct Graph\* graph = (struct Graph\*)malloc(sizeof(struct Graph));      graph->V = V;      // Create an array of adjacency lists. Size of array will be V      graph->array = (struct AdjList\*)malloc(V \* sizeof(struct AdjList));      // Initialize each adjacency list as empty by making head as NULL      for (int i = 0; i < V; ++i)          graph->array[i].head = NULL;      return graph;  }  void addEdge(struct Graph\* graph, int src, int dest, int weight) {      struct AdjListNode\* newNode = newAdjListNode(dest, weight);      newNode->next = graph->array[src].head;      graph->array[src].head = newNode;      newNode = newAdjListNode(src, weight);      newNode->next = graph->array[dest].head;      graph->array[dest].head = newNode;  }  int minKey(int key[], bool mstSet[], int V) {      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[], int n, struct AdjList\* array) {      printf("Edge   Weight\n");      for (int i = 1; i < n; i++) {          struct AdjListNode\* pCrawl = array[i].head;          while (pCrawl != NULL) {              if (pCrawl->dest == parent[i]) {                  printf("%d - %d    %d \n", parent[i], i, pCrawl->weight);                  break;              }              pCrawl = pCrawl->next;          }      }  }  void primMST(struct Graph\* graph) {      int V = graph->V;      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, V);          mstSet[u] = true;          struct AdjListNode\* pCrawl = graph->array[u].head;          while (pCrawl != NULL) {              int v = pCrawl->dest;              int weight = pCrawl->weight;              if (mstSet[v] == false && weight < key[v])                  parent[v] = u, key[v] = weight;              pCrawl = pCrawl->next;          }      }      // Print the constructed MST      printMST(parent, V, graph->array);  }  int main() {      int choice, V, E, src, dest, weight;        printf("Enter the number of vertices: ");      scanf("%d", &V);      struct Graph\* graph = createGraph(V);      printf("Enter the number of edges: ");      scanf("%d", &E);      printf("Enter source, destination and weight for each edge:\n");      for (int i = 0; i < E; ++i) {          scanf("%d %d %d", &src, &dest, &weight);          addEdge(graph, src, dest, weight);      }      printf("Minimum Spanning Tree using Prim's Algorithm:\n");      primMST(graph);      return 0;  } |

Output:

