NAME: **Ayush Sanjay Dhangar**

# ROLL NO: 42 CLASS: IT - A BATCH: 02

Problem Statement: Write C/C++ program for representation of graphs using adjacency matrix and adjacency lists

# PROGRAM:

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

struct Node { int dest;

struct Node\* next;

};

struct Graph { int V;

struct Node\*\* array;

};

struct Node\* createNode(int dest) {

struct Node\* newNode = (struct Node\*)malloc(sizeof(struct Node)); newNode->dest = dest;

newNode->next = NULL; return newNode;

}

struct Graph\* createGraph(int V) {

struct Graph\* graph = (struct Graph\*)malloc(sizeof(struct Graph)); graph->V = V;

graph->array = (struct Node\*\*)malloc((V + 1) \* sizeof(struct Node\*));

for (int i = 1; i <= V; ++i) graph->array[i] = NULL;

return graph;

}

void addEdge(struct Graph\* graph, int src, int dest) { struct Node\* newNode = createNode(dest);

newNode->next = graph->array[src]; graph->array[src] = newNode; newNode = createNode(src);

newNode->next = graph->array[dest]; graph->array[dest] = newNode;

}

void printGraph(struct Graph\* graph) { for (int v = 1; v <= graph->V; ++v) {

struct Node\* pCrawl = graph->array[v]; printf("Adjacency list of vertex %d\nhead ", v); while (pCrawl) {

printf("-> %d", pCrawl->dest); pCrawl = pCrawl->next;

}

printf("\n\n");

}

}

void adjacencyMatrix() { int N, M;

printf("Enter the number of vertices: "); scanf("%d", &N);

printf("Enter the number of edges: "); scanf("%d", &M);

int Adj[N + 1][N + 1];

for (int i = 0; i <= N; i++) {

for (int j = 0; j <= N; j++) { Adj[i][j] = 0;

}

}

printf("Enter the edges (source destination):\n"); for (int i = 0; i < M; i++) {

int x, y;

scanf("%d %d", &x, &y);

Adj[x][y] = 1;

Adj[y][x] = 1;

}

printf("\nGraph representation using adjacency matrix\n"); for (int i = 1; i <= N; i++) {

for (int j = 1; j <= N; j++) { printf("%d ", Adj[i][j]);

}

printf("\n");

}

}

void adjacencyList() { int V, E;

printf("Enter the number of vertices: "); scanf("%d", &V);

printf("Enter the number of edges: "); scanf("%d", &E);

struct Graph\* graph = createGraph(V);

printf("Enter the edges (source destination):\n"); for (int i = 0; i < E; ++i) {

int src, dest;

scanf("%d %d", &src, &dest);

addEdge(graph, src, dest);

}

printf("\nGraph representation using adjacency list\n"); printGraph(graph);

}

int main() { int choice;

while (1) {

printf("1. Graph Representation using Adjacency matrix\n"); printf("2. Graph Representation using Adjacency list\n"); printf("3. Exit\n");

printf("Enter choice: "); scanf("%d", &choice);

switch (choice) { case 1:

adjacencyMatrix(); break;

case 2:

adjacencyList(); break;

case 3:

exit(0); default:

printf("Invalid choice\n"); break;

}

}

return 0;

}

OUTPUT:

1. Graph Representation using Adjacency matrix
2. Graph Representation using Adjacency list
3. Exit

Enter choice: 1

Enter the number of vertices: 5 Enter the number of edges: 6

Enter the edges (source destination): 1 2

1 3

1 4

2 4

3 5

4 5

Graph representation using adjacency matrix 0 1 1 1 0

1 0 0 1 0

1 0 0 0 1

1 1 0 0 1

0 0 1 1 0

1. Graph Representation using Adjacency matrix
2. Graph Representation using Adjacency list
3. Exit

Enter choice: 2

Enter the number of vertices: 5 Enter the number of edges: 6

Enter the edges (source destination): 1 2

1 3

1 4

2 4

3 5

4 5

Graph representation using adjacency list

Adjacency list of vertex 1 head -> 4-> 3-> 2 Adjacency list of vertex 2 head -> 4-> 1

Adjacency list of vertex 3 head -> 5-> 1

Adjacency list of vertex 4 head -> 5-> 2-> 1 Adjacency list of vertex 5 head -> 4-> 3

1. Graph Representation using Adjacency matrix
2. Graph Representation using Adjacency list
3. Exit

Enter choice: 3