**DSC LAB – {6}**

Akshaya agarwal

2247207

3-MCA B

**CODE :**

#include <stdio.h>

#include <stdlib.h>

#define MAX\_VERTICES 100

int adj[MAX\_VERTICES][MAX\_VERTICES]; // adjacency matrix

int visited[MAX\_VERTICES]; // array to keep track of visited vertices

int queue[MAX\_VERTICES]; // array to implement BFS queue

int stack[MAX\_VERTICES]; // array to implement DFS stack

int rear = -1, front = -1; // variables to keep track of the BFS queue

int top = -1; // variable to keep track of the DFS stack

// function to add an edge between two vertices

void addEdge(int u, int v) {

adj[u][v] = 1;

adj[v][u] = 1;

}

// function to perform BFS on the graph

void bfs(int start, int n) {

int i;

visited[start] = 1;

queue[++rear] = start;

while (front != rear) {

int current = queue[++front];

printf("%d ", current);

for (i = 0; i < n; i++) {

if (adj[current][i] == 1 && visited[i] == 0) {

visited[i] = 1;

queue[++rear] = i;

}

}

}

}

// function to perform DFS on the graph

void dfs(int start, int n) {

int i;

visited[start] = 1;

stack[++top] = start;

printf("%d ", start);

while (top != -1) {

int current = stack[top];

int found = 0;

for (i = 0; i < n; i++) {

if (adj[current][i] == 1 && visited[i] == 0) {

visited[i] = 1;

stack[++top] = i;

printf("%d ", i);

found = 1;

break;

}

}

if (found == 0) {

top--;

}

}

}

int main() {

int i, j, n, option, u, v, start;

printf("\nBuilding an adjacency matrix to print BFS and DFS\n\n");

printf("Enter the number of vertices: ");

scanf("%d", &n);

while (1) {

printf("\nMenu\n");

printf("1. Add an edge\n");

printf("2. BFS traversal\n");

printf("3. DFS traversal\n");

printf("4. Exit\n");

printf("Enter your option: ");

scanf("%d", &option);

switch (option) {

case 1:

printf("Enter the vertices (u v): ");

scanf("%d %d", &u, &v);

addEdge(u, v);

break;

case 2:

printf("Enter the starting vertex: ");

scanf("%d", &start);

for (i = 0; i < n; i++) {

visited[i] = 0;

}

bfs(start, n);

break;

case 3:

printf("Enter the starting vertex: ");

scanf("%d", &start);

for (i = 0; i < n; i++) {

visited[i] = 0;

}

dfs(start, n);

break;

case 4:

exit(0);

default:

printf("Invalid option\n");

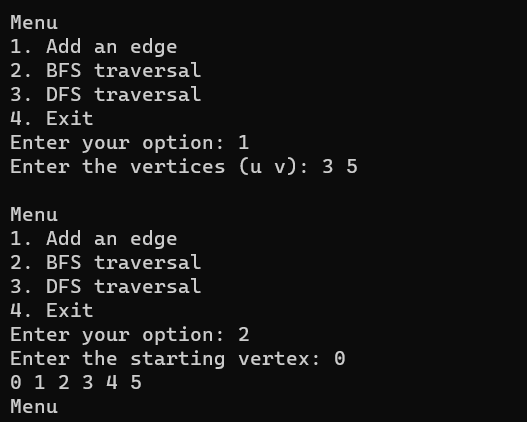
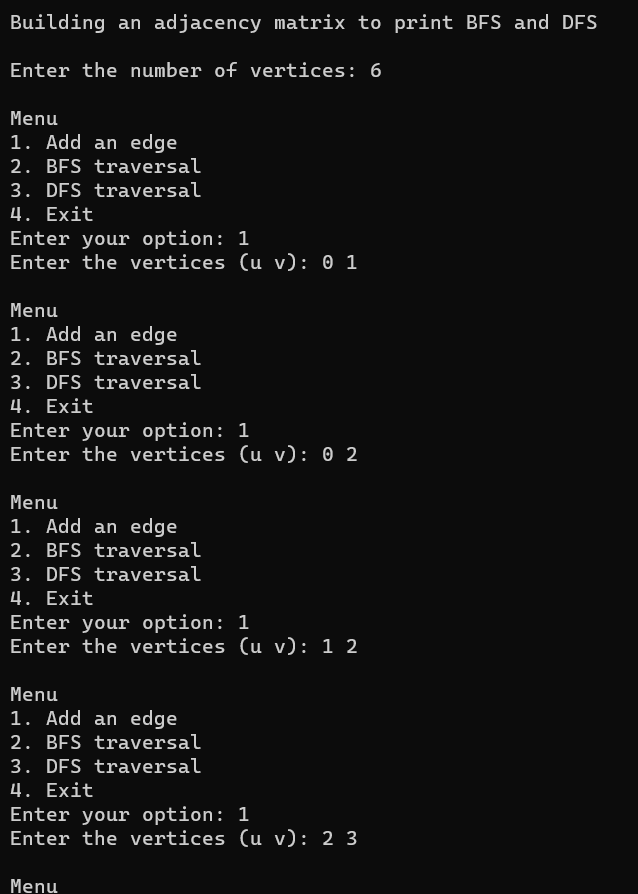
}

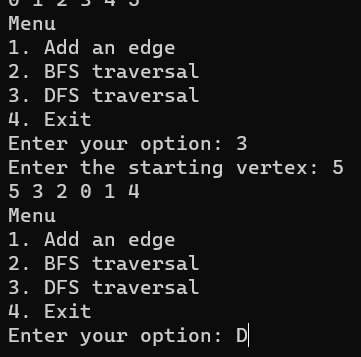
}

return 0;

}

**OUTPUT :**

****

****