DAY 14 – GRAPHS

17. Write a menu driven C program to perform the following operations on a directed graph

(i) In degree of a particular node

(ii) Out degree of a particular node

(iii) DFS

(iv) BFS

(v) Display (using Adjacency List and Adjacency Matrix).

PROGRAM

```
#include<stdio.h>
#include<stdlib.h>
#define MAX 100
struct vertex
{
    int value;
    int status;
    struct vertex *next;
}*vertices_list[MAX], *a[MAX];
int n, vertices_array[MAX], adj_matrix[MAX][MAX];
int top = -1, front = -1, rear = -1;
void display_adj_matrix()
{
    int i, j;
    printf("\nAdjaceny Matrix -\n");
    for(i = 0; i < n; i++)
    {
        for(j = 0; j < n; j++)
        printf("%d\t", adj_matrix[i][j]);
        printf("\n");
    }
}
void display_adj_list()
{
    int i, j;
    struct vertex *ptr;
    printf("\nAdjacency List -\n");
    for(i = 0; i < n; i++)
    {
        ptr = vertices_list[i];
        while(ptr -> next != NULL)
        {
            printf("%d -> ", ptr -> value);
            ptr=ptr->next;
```

```
        }
        printf("%d -> ", ptr -> value);
        if(ptr -> next == NULL)
            printf("NULL");
        printf("\n");
    }
}
void indegree()
{
    int i, j, item, flag = 0, count = 0;
    printf("Enter vertex: ");
    scanf("%d", &item);
    for(i = 0; i < n; i++)
    {
        if(vertices_array[i] == item)
        {
            flag = 1;
            break;
        }
    }
    if(flag == 0)
    {
        printf("Vertex not found.");
        return;
    }
    for(j = 0; j < n; j++)
        count += adj_matrix[j][i];
    printf("In-degree of %d: %d", item, count);
}
void outdegree()
{
    int i, j, item, flag = 0, count = 0;
    printf("Enter vertex: ");
    scanf("%d", &item);
    for(i = 0; i < n; i++)
    {
        if(vertices_array[i] == item)
        {
            flag = 1;
            break;
        }
    }
    if(flag == 0)
    {
        printf("Vertex not found.");
        return;
    }
    for(j = 0; j < n; j++)
        count += adj_matrix[i][j];
    printf("Out-degree of %d: %d", item, count);
}
void enqueue(struct vertex *item)
{
    if(rear == MAX -1)
```

```
    {
        printf("Queue overflow.\n");
        exit(0);
    }
    a[++rear] = item;
    if(front == -1)
    front = 0;
}
struct vertex * dequeue()
{
    struct vertex *item;
    item = a[front];
    if(front == rear)
    front = rear = -1;
    else
    front++;
    return item;
}
void bfs()
{
    int i, item, flag = 0;
    struct vertex *ptr;
    printf("Enter starting vertex: ");
    scanf("%d", &item);
    for(i = 0; i < n; i++)
    vertices_list[i] -> status = 1;
    for(i = 0; i < n; i++)
    {
        if(vertices_list[i] -> value == item)
        {
            flag = 1;
            break;
        }
    }
    if(flag == 0)
    {
        printf("Vertex not found.");
        return;
    }
    enqueue(vertices_list[i]);
    vertices_list[i] -> status = 2;
    printf("BFS: ");
    while(front != -1)
    {
        ptr = dequeue();
        printf("%d ", ptr -> value);
        ptr -> status = 3;
        while(ptr -> next != NULL)
        {
            ptr = ptr -> next;
            for(i = 0; i < n; i++)
            {
                if(vertices_list[i] -> value == ptr -> value)
                break;
```

```
            }
            if(vertices_list[i] -> status == 1)
            {
                enqueue(vertices_list[i]);
                vertices_list[i] -> status = 2;
            }
        }
    }
}
void push(struct vertex *item)
{
    if(top == MAX - 1)
    {
        printf("Stack overflow.");
        exit(0);
    }
    a[++top] = item;
}
struct vertex * pop()
{
    return a[top--];
}
void dfs()
{
    int i, item, flag = 0;
    struct vertex *ptr;
    printf("Enter starting vertex: ");
    scanf("%d", &item);
    for(i = 0; i < n; i++)
    vertices_list[i] -> status = 1;
    for(i = 0; i < n; i++)
    {
        if(vertices_list[i] -> value == item)
        {
            flag = 1;
            break;
        }
    }
    if(flag == 0)
    {
        printf("Vertex not found.");
        return;
    }
    push(vertices_list[i]);
    vertices_list[i] -> status = 2;
    printf("DFS: ");
    while(top != -1)
    {
        ptr = pop();
        printf("%d ", ptr -> value);
        ptr -> status = 3;
        while(ptr -> next != NULL)
        {
            ptr = ptr -> next;
```

```
            for(i = 0; i < n; i++)
            {
                if(vertices_list[i] -> value == ptr -> value)
                break;
            }
            if(vertices_list[i] -> status == 1)
            {
                push(vertices_list[i]);
                vertices_list[i] -> status = 2;
            }
        }
    }
}
void main()
{
    int i, j, choice, menu;
    struct vertex *newvertex, *ptr;
    printf("Enter no. of vertices: ");
    scanf("%d", &n);
    printf("Enter the vertices: ");
    for(i = 0; i < n; i++)
    {
        scanf("%d", &vertices_array[i]);
        newvertex = (struct vertex *)malloc(sizeof(struct vertex));
        newvertex -> value = vertices_array[i];
        newvertex -> next = NULL;
        vertices_list[i] = newvertex;
    }
    for(i = 0; i < n; i++)
    {
        for(j = 0; j < n; j++)
        {
            printf("Is an edge present from %d to %d? - 1. Yes, 2. No: ", vertices_array[i],
vertices_array[j]);
            scanf("%d", &choice);
            if(choice == 1)
            {
                adj_matrix[i][j] = 1;
                newvertex = (struct vertex *)malloc(sizeof(struct vertex));
                newvertex -> value = vertices_array[j];
                newvertex -> next = NULL;
                ptr = vertices_list[i];
                while(ptr -> next != NULL)
                ptr = ptr -> next;
                ptr -> next = newvertex;
            }
            else
                adj_matrix[i][j] = 0;
        }
    }
    do
    {
        printf("\n\t\t\tMENU");
        printf("\n1. In-degree\t2. Out-degree\t\t3. BFS");
```

```
        printf("\n4.DFS\t\t5. Adjacency Matrix\t6. Adjacency List");
        printf("\nEnter choice: ");
        scanf("%d", &menu);
        switch(menu)
        {
        case 1: indegree();
                break;
        case 2: outdegree();
                break;
        case 3: bfs();
                break;
        case 4: dfs();
                break;
        case 5: display_adj_matrix();
                break;
        case 6: display_adj_list();
                break;
        }
    }while(menu >= 1 && menu <= 6);
}
```

OUTPUT

```
Enter no. of vertices: 6
Enter the vertices: 0 1 2 3 4 5
Is an edge present from 0 to 0? - 1. Yes, 2. No: 2
Is an edge present from 0 to 1? - 1. Yes, 2. No: 1
Is an edge present from 0 to 2? - 1. Yes, 2. No: 1
Is an edge present from 0 to 3? - 1. Yes, 2. No: 1
Is an edge present from 0 to 4? - 1. Yes, 2. No: 2
Is an edge present from 0 to 5? - 1. Yes, 2. No: 2
Is an edge present from 1 to 0? - 1. Yes, 2. No: 2
Is an edge present from 1 to 1? - 1. Yes, 2. No: 2
Is an edge present from 1 to 2? - 1. Yes, 2. No: 1
Is an edge present from 1 to 3? - 1. Yes, 2. No: 2
Is an edge present from 1 to 4? - 1. Yes, 2. No: 2
Is an edge present from 1 to 5? - 1. Yes, 2. No: 1
Is an edge present from 2 to 0? - 1. Yes, 2. No: 2
Is an edge present from 2 to 1? - 1. Yes, 2. No: 2
Is an edge present from 2 to 2? - 1. Yes, 2. No: 2
Is an edge present from 2 to 3? - 1. Yes, 2. No: 2
Is an edge present from 2 to 4? - 1. Yes, 2. No: 2
Is an edge present from 2 to 5? - 1. Yes, 2. No: 1
Is an edge present from 3 to 0? - 1. Yes, 2. No: 2
Is an edge present from 3 to 1? - 1. Yes, 2. No: 2
Is an edge present from 3 to 2? - 1. Yes, 2. No: 2
Is an edge present from 3 to 3? - 1. Yes, 2. No: 2
Is an edge present from 3 to 4? - 1. Yes, 2. No: 1
Is an edge present from 3 to 5? - 1. Yes, 2. No: 2
Is an edge present from 4 to 0? - 1. Yes, 2. No: 1
Is an edge present from 4 to 1? - 1. Yes, 2. No: 2
Is an edge present from 4 to 2? - 1. Yes, 2. No: 2
Is an edge present from 4 to 3? - 1. Yes, 2. No: 2
Is an edge present from 4 to 4? - 1. Yes, 2. No: 2
Is an edge present from 4 to 5? - 1. Yes, 2. No: 2
Is an edge present from 5 to 0? - 1. Yes, 2. No: 2
Is an edge present from 5 to 1? - 1. Yes, 2. No: 2
Is an edge present from 5 to 2? - 1. Yes, 2. No: 2
Is an edge present from 5 to 3? - 1. Yes, 2. No: 2
Is an edge present from 5 to 4? - 1. Yes, 2. No: 2
Is an edge present from 5 to 5? - 1. Yes, 2. No: 2
```

```
                    MENU
1. In-degree    2. Out-degree           3. BFS
4.DFS           5. Adjacency Matrix     6. Adjacency List
Enter choice: 1
Enter vertex: 5
In-degree of 5: 2
                    MENU
1. In-degree    2. Out-degree           3. BFS
4.DFS           5. Adjacency Matrix     6. Adjacency List
Enter choice: 2
Enter vertex: 0
Out-degree of 0: 3
                    MENU
1. In-degree    2. Out-degree           3. BFS
4.DFS           5. Adjacency Matrix     6. Adjacency List
Enter choice: 3
Enter starting vertex: 0
BFS: 0 1 2 3 5 4
                    MENU
1. In-degree    2. Out-degree           3. BFS
4.DFS           5. Adjacency Matrix     6. Adjacency List
Enter choice: 4
Enter starting vertex: 3
DFS: 3 4 0 2 5 1
                    MENU
1. In-degree    2. Out-degree           3. BFS
4.DFS           5. Adjacency Matrix     6. Adjacency List
Enter choice: 5

Adjaceny Matrix -
0       1       1       1       0       0
0       0       1       0       0       1
0       0       0       0       0       1
0       0       0       0       1       0
1       0       0       0       0       0
0       0       0       0       0       0
```

```
                    MENU
1. In-degree    2. Out-degree           3. BFS
4.DFS           5. Adjacency Matrix     6. Adjacency List
Enter choice: 6

Adjacency List -
0 -> 1 -> 2 -> 3 -> NULL
1 -> 2 -> 5 -> NULL
2 -> 5 -> NULL
3 -> 4 -> NULL
4 -> 0 -> NULL
5 -> NULL
```