Aim:- To study Implementation of BFS and DFS on a directed graph using adjacency matrix

Program : BFS using Adjacency

Matrix Code:

```
#include <stdio.h>
#include <stdlib.h>
int vertex = 5;
int delete;
int adjacencyMatrix[20][20] = {
    {0, 1, 1, 1, 0},
    {1, 0, 1, 0, 0},
    {1, 1, 0, 0, 1},
    {1, 0, 0, 0, 0},
    {0, 0, 1, 0, 0}
};
int queue[20], front = -1, rear = -1;
// for traversal
int visited[20];    // for printing and preventing
repetition int deleted;
int indexVisited = 0;    // index of visited array
// for queue
void insert(int item)
{     if (front == -1)
{
    front++;
    }
```

```
    rear++;     queue[rear] = item;
}
// for queue
void del(int* deleted)
{     if (front==-1 || front>rear) {         return;
    }
    *deleted = queue[front];
    front++;
}

int isPresentVisited(int num) {
    int i;
    for (i = 0; i < indexVisited ; i++) {         if (visited[i]
== num) {             return 1;
        } }
    return 0;
}

int isPresentQueue(int num) {    // check presence of
number in queue
    for (int i = front; i <= rear; i++) {         if (queue[i]
== num) {             return 1;
        }
    }
    return 0;
}
void bfs(int start, int vertex) {
    // initialization
```

```
    visited[indexVisited++] = start;
    int i = 0;
    while (i < vertex) {         if (adjacencyMatrix[start][i]
&& !isPresentVisited(i)) {
            insert(i);
        }
        i++;
    }
    while (front <= rear)
    {
    del(&delete);
    visited[indexVisited++] = delete;
        for (i = 0; i < vertex; i++)
        {             if (adjacencyMatrix[delete][i] &&
!isPresentVisited(i) && !isPresentQueue(i))
        {
                insert(i);
            }
        }
        printf("\n");
    }
}
int main()
{
    printf("D10A_Atharva Chavan_9");
    bfs(0, vertex);
    for (int i = 0; i < vertex; i++)
    {         printf("%d\t", visited[i]);
```

```
    }
}
```

Output:-

```
/tmp/mzO9gGyZGf.o
D10A_Atharva Chavan_9


0    1    2    3    4
```

Program : DFS using Adjacency

Matrix Code:

```
#include <stdio.h>
#include <stdlib.h>
#define MAX 50  //Max size of stack

int popped; int stack[MAX]; //Defining stack
int vertex = 6;
int AdjacencyMatrix[20][20] = {
    {0, 1, 1, 1, 0, 0},
    {1, 0, 0, 1, 1, 0},
    {1, 0, 0, 1, 0, 1},
    {1, 1, 1, 0, 1, 1},
    {0, 1, 0, 1, 0, 1},
    {0, 0, 1, 1, 1, 0}
};
```

```
int adjacencyMatrix[20][20] = {
    {0, 1, 1, 1, 0},
    {1, 0, 1, 0, 0},
    {1, 1, 0, 0, 1},
    {1, 0, 0, 0, 0},
    {0, 0, 1, 0, 0}
};

int top = -1; int visited[10]; // for printing and
preventing repetition
int deleted;
int indexVisited = 0;    // index of visited array

void push(int elem) {
    top++;     stack[top] = elem;
}

void pop(int *popped) {     *popped = stack[top];
    top--;
}

int isPresentVisited(int num) {    // check presence of
number in visited
    int i;
    for (i = 0; i < indexVisited ; i++) {         if (visited[i]
== num) {             return 1;
        }
    }
```

```
    return 0;
}

void dfs(int start, int vertex)
{     push(start);
visited[indexVisited++] = start;

    while (top>=0)
    {    for (int i = 0; i < vertex; i++) {
            if (AdjacencyMatrix[stack[top]][i] &&
!isPresentVisited(i)) {                 visited[indexVisited++]
= i;
            push(i);
            break;
            }
            if (i == vertex-1)
            {   pop(&popped);
            }
        }
    }
}

int main() {
printf("D10A_Atharva Chavan_9\n");
printf("\n");
dfs(2, vertex);
for (int i = 0; i < vertex; i++)
{         printf("%d\t", visited[i]);
```

```
    }

}
```

Output:-

```
/tmp/mzO9gGyZGf.o
D10A_Atharva Chavan_9

2    0    1    3    4    5
```