```
// S Harshini-185001058

#include<stdio.h>
#include<stdlib.h>
#include"dfs.h"
#include"bfs.h"

int main(){

	char adj[10][10];
	char adj1[10][10];
	int no_of_vertices,i,j;
	char c;
	printf("\nEnter no of vertices:");
	scanf("%d",&no_of_vertices);
	printf("\n\nEnter vertices notations ");adj[0][0]=0;
	for(i=1;i<=no_of_vertices;i++){
		printf("\n");
		scanf(" %c",&c);
		adj[i][0]=adj[0][i]=c;
		adj1[i][0]=adj1[0][i]=c;
	}
	char start=adj[1][0];
	adj[0][no_of_vertices+1]=adj1[0][no_of_vertices+1]='V';
	printf("\nEnter the adjacency vectors for the Vertices:");
	for(i=1;i<=no_of_vertices;i++){
		printf("\nVertice %c : ",adj[i][0]);
		for(j=1;j<=no_of_vertices;j++){
			scanf(" %c",&adj[i][j]);adj1[i][j]=adj[i][j];
		}
		adj[i][no_of_vertices+1]=adj1[i][no_of_vertices+1]='0';
	}
	DFS(adj,no_of_vertices,start);
	BFS(adj1,no_of_vertices,start);



	return 0;
}

/* SAMPLE INPUT/OUTPUT
Enter no of vertices:5
```

```
Enter vertices notations
0

1

2

3

4

Enter the adjacency vectors for the Vertices:
Vertice 0 : 0 1 0 0 0

Vertice 1 : 0 0 1 0 0

Vertice 2 : 0 0 0 1 1

Vertice 3 : 1 0 0 0 0

Vertice 4 : 0 0 1 0 0

depth first traversal
0,1,2,4,3,
Breadth first traversal
0,1,2,3,4,
*/


/************ contents of dfs file**************************/
        typedef struct node{
        char name;
        struct node *next;
}node;

typedef struct stack{
        node *top;
}stack;


void push(stack *s,node *p){
        if(s->top==NULL){
```

```
                s->top=p;
                return;
        }
        p->next=s->top;
        s->top=p;
}

char pop(stack *s){
        char temp=s->top->name;
        s->top=s->top->next;
        return(temp);
}



void DFS(char adj[][10],int no_of_vertices,char start){

        stack *s=(stack *)malloc(sizeof(stack));
        s->top=NULL;

        node *t;
        char T;
        int count=0,i,j;
        t=(node*)malloc(sizeof(node));
        t->name=start;
        t->next=NULL;
        push(s,t);
        printf("\ndepth first traversal \n");
        do{
        T=pop(s);
        printf("%c,",T);
        count++;
        for(i=0;i<=no_of_vertices;i++){
                if(adj[i][0]==T){
                        adj[i][no_of_vertices+1]='1';
                }
        }
        for(i=1;i<=no_of_vertices;i++){
                if(adj[i][0]==T){
                        for(j=1;j<=no_of_vertices;j++){
                                if(adj[i][j]=='1' && adj[j][no_of_vertices+1]=='0'){
                                node *t=(node*)malloc(sizeof(node));
                                t->name=adj[0][j];
```

```
                                t->next=NULL;
                                push(s,t);
                                }
                        }
                        break;
                }
        }

        }while(count<no_of_vertices);

}


/*****************contents of bfs file*****************************/
typedef struct queue{
        node *front;
        node *rear;
}queue;


void enqueue(queue *q,node *p){
        if(q->front==q->rear && q->rear==NULL){
                q->front=q->rear=p;
                return;
        }
        q->rear->next=p;
        q->rear=p;
}

char dequeue(queue *q){
        char temp=q->front->name;
        if(q->front==q->rear){
                q->front=q->rear=NULL;return temp;
        }
        q->front=q->front->next;
        return(temp);
}


void BFS(char adj[][10],int no_of_vertices,char start){
        char T;
        node *t;
```

```
        int count=0,i,j;
        printf("\nBreadth first traversal\n");

        queue *q=(queue *)malloc(sizeof(queue));
        q->front=q->rear=NULL;
        t=(node*)malloc(sizeof(node));
        t->name=start;
        t->next=NULL;
        enqueue(q,t);
        do{
        T=dequeue(q);
        printf("%c,",T);
        count++;
        for(i=0;i<=no_of_vertices;i++){
                if(adj[i][0]==T){
                        adj[i][no_of_vertices+1]='1';
                }
        }
        for(i=1;i<=no_of_vertices;i++){
                if(adj[i][0]==T){
                        for(j=1;j<=no_of_vertices;j++){
                                if(adj[i][j]=='1' && adj[j][no_of_vertices+1]=='0'){
                                node *t=(node*)malloc(sizeof(node));
                                t->name=adj[0][j];
                                t->next=NULL;
                                enqueue(q,t);
                                }
                        }
                        break;
                }
        }

        }while(count<no_of_vertices);

}
```