SSN COLLEGE OF ENGINEERING

DEPARTMENT OF CSE

ASSIGNMENT11

NAME:S.MADHUMITHA

ROLLNO:185001086

CODE:

FILE1:DFS

```
#include<stdio.h>
#include<stdlib.h>
#define max 10
#define SIZE 5
#define MAX 5

char s[SIZE];
int top = -1;
struct sNode
{
char data;
struct sNode *next;
};
void push(char elem) {

 s[++top] = elem;

}

char pop() {
```

```
 return (s[top--]);

}
void dfs(int mat[][max],int no,int pos){
    int i,j,k=0;
    char vis[max];int f;
    for(i=0;i<no;i++)vis[i]=' ';
    char arr[26],val;
    for(i=0;i<26;i++){
        arr[i]=(char)(i+(int)('a'));
    }
    push(arr[pos]);
    while(top!=-1){
        val=pop();
        for(i=0;i<26;i++){if(val==arr[i])pos=i;}

        for(j=0;j<no;j++){
            if(val!=vis[j])f=0;
            else f=1;
    }
    if(f==0)vis[k++]=val;
    for(i=no-1;i>-1;i--){
        if(mat[pos][i]==1 && vis[i]==' '){
            push(arr[i]);
        }
    }
```

```
    }
    for(i=0;i<no;i++)printf("%c",vis[i]);
    return;
}
```

FILE2:BFS

```
#include<stdio.h>
#include<stdlib.h>
#define max 10
#define SIZE 5
#define MAX 5

char queue_array[MAX];
int front = -1;
int rear = -1;

void enqueue(int element)
{
        if(isFull())
        {
                printf("\nQueue Overflow\n");
                return;
        }
        else if(front == -1)
        {
                front = 0;
        }
```

```
        rear = rear + 1;
        queue_array[rear] = element;
}

char dequeue()
{
        int item;
        if(isEmpty())
        {
                printf("\nQueue Underflow\n");
                return 1;
        }
        else
        {
                item = queue_array[front];
                front = front + 1;
                return item;
        }
}
int isFull()
{
        if(rear == MAX - 1)
                return 1;
        else
                return 0;
}
```

```
int isEmpty()
{
        if(front == -1 || front == rear + 1)
                return 1;
        else
                return 0;
}
void bfs(int mat[][max],int no,int pos){
    char arr[26],val;int f=0,k=0,i,j;
    char vis[max];
    for(i=0;i<no;i++)vis[i]=' ';
    for(i=0;i<26;i++){
        arr[i]=(char)(i+(int)('a'));
    }
    enqueue(arr[pos]);
    while(!isEmpty()){
        val=dequeue();//printf("%c",val);
        for(i=0;i<26;i++){if(val==arr[i])pos=i;}
        for(i=0;i<no;i++){if(val!=vis[i])f=1;
        else f=0;}
        if(f==1)vis[k++]=val;
        for(j=0;j<no;j++){
            if(mat[pos][j]==1 && vis[j]==' '){
                enqueue(arr[j]);
            }
        }
```

```
    }
    printf("BFS TRAVERSAL\n");
    for(i=0;i<no;i++)printf("%c",vis[i]);

}
```

FILE3:GRAPHS

```
#include<stdio.h>
#include<stdlib.h>
#define max 10
#define SIZE 5
#define MAX 5
#include “dfs.h”
#include “bfs.h”

typedef struct graph{
    int size,capcity;
    int adj[max][max];
}graph;
void insertnode(int mat[][max],char n,int no){
    int i,pos;int c=1;char node;
    char arr[26];
    for(i=0;i<26;i++){
        arr[i]=(char)(i+(int)('a'));
    }
    for(i=0;i<no;i++){
```

```
        if(n==arr[i]){pos=i;break;}
    }
    printf("enter the adjacent nodes of %c",n);
    while(c==1){
        printf("adj node:");
        scanf(" %c",&node);
        for(i=0;i<no;i++){
            if(node==arr[i]){
                mat[pos][i]=1;
            }

        }
                printf("1/0 (add/exit)");scanf("%d",&c);
    }

    return;
}
int main(){
    graph g;
    int e=5,i,j,num;char nodes;

    printf("enter no of nodes");
    scanf("%d",&num);
    for(i=0;i<num;i++){
        for(j=0;j<num;j++){
            g.adj[i][j]=0;
        }
```

```
    }


    for(i=0;i<num;i++){
        printf("node:");scanf(" %c",&nodes);
        insertnode(g.adj,nodes,num);
    }
    /*for(i=0;i<num;i++){
        for(j=0;j<num;j++){
            printf("%d",g.adj[i][j]);
        }printf("\n");
    }*/
    printf("DFS TRAVERSAL");
    dfs(g.adj,num,0);
    bfs(g.adj,num,0);
return 0;
}
```

OUTPUT1:

```
enter no of nodes5
node:a
enter the adjacent nodes of aadj node:b
1/0 (add/exit)1
```

adj node:c

1/0 (add/exit)1

adj node:e

1/0 (add/exit)0

node:b

enter the adjacent nodes of badj node:d

1/0 (add/exit)1

adj node:e

1/0 (add/exit)0

node:c

enter the adjacent nodes of cadj node:a

1/0 (add/exit)0

node:d

enter the adjacent nodes of dadj node:b

1/0 (add/exit)0

node:e

enter the adjacent nodes of eadj node:a

1/0 (add/exit)2 1

adj node:b

1/0 (add/exit)0

DFS TRAVERSAL abdec

BFS TRAVERSAL

abced

OUTPUT2:

enter no of nodes5

node:0

enter the adjacent nodes of aadj node:1

1/0 (add/exit)0

node:1

enter the adjacent nodes of badj node:2

1/0 (add/exit)0

node:2

enter the adjacent nodes of cadj node:3

1/0 (add/exit)1

adj node:4

1/0 (add/exit)0

node:3

enter the adjacent nodes of dadj node:0

1/0 (add/exit)0

node:4

enter the adjacent nodes of eadj node:2

1/0 (add/exit)0

DFS TRAVERSAL 01234

BFS TRAVERSAL

01234