Programmer Name: Param Chordiya
Batch: E5

Problem Statement:
Write a program to Implement Graph using adjacency Matrix, apply following traversal
1. Breadth First Search (BFS)
2. Depth First Search (DFS)

INPUT:

```
#include<stdio.h>
#include<stdlib.h>
typedef struct BSTnode
{
                    int info;
                    struct BSTnode *left,*right;
}BSTnode;
BSTnode *find(BSTnode *,int);
BSTnode *insert(BSTnode *,int);
BSTnode *create();
void inorder(BSTnode *T);
void preorder(BSTnode *T);
void postorder(BSTnode *T);
void main()
{
                    BSTnode *root=NULL,*p;
                    int x, op;
                    printf("\n**************************************\n");
                    printf("\t    ROLL NO:22119\n");
                    printf("**************************************\n");
                    printf("\t       MENU");
                    printf("\n**************************************\n");
                    do{

                    printf("1)Create\n2)Search\n3)Insert\n4)Inorder\n5)Preorder\n6)Pos
torder\n7)Exit\n");
                        printf("\nEnter your choice:");
```

```
            scanf("%d",&op);
            switch(op)
            {
                case 1:
                    root=create();
                    break;
                case 2:
                    printf("\nEnter the key to be searched :");
                    scanf("%d",&x);
                    p=find(root,x);
                    if(p=NULL)
                        printf("\n NOT FOUND\n");
                    else
                        printf("\n FOUND\n");
                        break;
                case 3:
                    printf("\Enter a Info to be inserted:");
                    scanf("%d",&x);
                    root=insert(root,x);
                    break;
                case 4:
                    preorder(root);
                    break;
                case 5:
                inorder(root);
                break;
                case 6:
                postorder(root);
                break;
            }
        }while(op!=7);
}

void inorder(BSTnode *T)
{
```

```
                    if(T!=NULL)
                    {
                        inorder(T->left);
                        printf("%d\t",T->info);
                        inorder(T->right);

                    }
}
void preorder(BSTnode *T)
{
                    if(T!=NULL)
                    {
                        printf("%d\t",T->info);
                        preorder(T->left);
                        preorder(T->right);
                    }
}

void postorder(BSTnode *T)
{
                    if(T!=NULL)
                    {
                        printf("%d\t",T->info);
                        postorder(T->left);
                        postorder(T->right);

                    }
}
BSTnode *find(BSTnode *root,int x)
{
                    while(root!=NULL)
                    {
                        if(x==root->info)
                                return(root);
                        if(x>root->info)
```

```
                        root=root->right;
                    else
                        root=root->left;
                }return(NULL);
}
BSTnode *insert(BSTnode *T,int x)
{
                    BSTnode *p,*q,*r;
                    r=(BSTnode*)malloc(sizeof(BSTnode));
                    r->info=x;
                    r->left=NULL;
                    r->right=NULL;
                    if(T==NULL)
                        return(r);
                        p=T;
                    while(p!=NULL)
                    {
                        q=p;
                        if(x>=p->info)
                            p=p->right;
                        else
                        if(x<p->info)
                            p=p->left;

                    }
                    if(x>=q->info)
                        q->right=r;
                    else
                        q->left=r;
                    return(T);
}
BSTnode *create()
{
                    int n,x,i;
                    BSTnode *root;
```

```
    root=NULL;
    printf("\nEnter no.of nodes :");
    scanf("%d",&n);
    printf("\nEnter the values :");
    for(i=0;i<n;i++)
    {
        scanf("%d",&x);
        root=insert(root,x);
    }
    return(root);

}
```

OUTPUT:

```
**************************************
            ROLL NO:22119
**************************************
                MENU
**************************************

1)Create
2)Search
3)Insert
4)Inorder
5)Preorder
6)Postorder
7)Exit

Enter your choice:1

Enter no.of nodes :2

Enter the values :9
5

1)Create
2)Search
3)Insert
4)Inorder
5)Preorder
6)Postorder
7)Exit

Enter your choice:2

Enter the key to be searched :5

 FOUND

1)Create
2)Search
3)Insert
4)Inorder
5)Preorder
6)Postorder
7)Exit

Enter your choice:3
enter a Info to be inserted:7
```

```
1)Create
2)Search
3)Insert
4)Inorder
5)Preorder
6)Postorder
7)Exit

Enter your choice:4
9       5       7
1)Create
2)Search
3)Insert
4)Inorder
5)Preorder
6)Postorder
7)Exit

Enter your choice:5
5       7       9
1)Create
2)Search
3)Insert
4)Inorder
5)Preorder
6)Postorder
7)Exit

Enter your choice:6
9       5       7
1)Create
2)Search
3)Insert
4)Inorder
5)Preorder
6)Postorder
7)Exit

Enter your choice:7

--------------------------------
Process exited after 32.07 seconds with return value 7
Press any key to continue . . .
```