PROGRAM 1

```
#include<stdio.h>
#include<string.h>

#define MAX 20

int top = -1;
char stack[MAX];

char push(char item)
{
        if(top == (MAX-1))
                printf("Stack Overflow\n");
        else
                stack[++top] =item;
}
char pop()
{
        if(top == -1)
                printf("Stack Underflow\n");
        else
                return stack[top--];
}

main()
{
        char str[20];
        int i;
        printf("Enter the string : " );
        gets(str);
        for(i=0;i<strlen(str);i++)
                push(str[i]);
```

```
        for(i=0;i<strlen(str);i++)
                str[i]=pop();
        printf("Reversed string is : ");
        puts(str);
}
```

PROGRAM 2

```
#include<stdio.h>
char stack[20];
int top = -1;
void push(char x)
{
    stack[++top] = x;
}

char pop()
{
    if(top == -1)
        return -1;
    else
        return stack[top--];
}

int priority(char x)
{
    if(x == '(')
        return 0;
    if(x == '+' || x == '-')
        return 1;
    if(x == '*' || x == '/')
        return 2;
}
```

```
main()
{
    char exp[20];
    char *e, x;
    printf("Enter the expression :: ");
    scanf("%s",exp);
    e = exp;
    while(*e != '\0')
    {
        if(isalnum(*e))
            printf("%c",*e);
        else if(*e == '(')
            push(*e);
        else if(*e == ')')
        {
            while((x = pop()) != '(')
                printf("%c", x);
        }
        else
        {
            while(priority(stack[top]) >= priority(*e))
                printf("%c",pop());
            push(*e);
        }
        e++;
    }
    while(top != -1)
    {
        printf("%c",pop());
    }
}
```

PROGRAM 3

```
/*DEPTH FIRST SEARCH*/
#include<stdio.h>
void DFS(int);
int G[10][10],visited[10]={0},n;
void main()
{
int i,j;
printf("Enter number of vertices: ");
scanf("%d",&n);
printf("\nEnter adjacency matrix of Graph : ");
for(i=0;i<n;i++)
{
for(j=0;j,n;j++)
scanf("%d",&G[i][j]);
}
DFS(0);
}
void DFS(int i)
{
int j;
printf("\n%d",i);
visited[i]=1;
for(j=0;j<n;j++)
if(!visited[j]&&G[i][j]==1)
DFS(j);
}
```

PROGRAM 4

```
/*BREADTH FIRST SEARCH*/
#include<stdio.h>
int S[20][20],q[20]={0},n,visited[20]={0},i,j,f=0,r=-1;
void BFS(int v)
{
for(i=0;i<n;i++)
if(S[v][i]&&visited[i]==0)
q[++r]=i;
if(f<=r)
{
visited[q[f]]=1;
BFS(q[f++]);
}
}
```

PROGRAM 5

```
#include <stdio.h>

#include <stdlib.h>
```

```
struct node
{
    int data;
    struct node *next;
};
void push(struct node** top, int data);
int pop(struct node** top);
struct queue
{
    struct node *stack1;
    struct node *stack2;
};
void enqueue(struct queue *q, int x)
{
    push(&q->stack1, x);
}
void dequeue(struct queue *q)
{
    int x;
    if (q->stack1 == NULL && q->stack2 == NULL) {
        printf("queue is empty");
        return;
    }
    if (q->stack2 == NULL) {
        while (q->stack1 != NULL) {
        x = pop(&q->stack1);
        push(&q->stack2, x);
        }
    }
    x = pop(&q->stack2);
    printf("%d\n", x);
}
```

```
void push(struct node** top, int data)
{
    struct node* newnode = (struct node*) malloc(sizeof(struct node));
        if (newnode == NULL) {
            printf("Stack overflow \n");
            return;
        }
    newnode->data = data;
    newnode->next = (*top);
    (*top) = newnode;
}
int pop(struct node** top)
{
    int buff;
    struct node *t;
    if (*top == NULL) {
        printf("Stack underflow \n");
        return;
    }
    else {
        t = *top;
        buff = t->data;
        *top = t->next;
        free(t);
        return buff;
    }
}
void display(struct node *top1,struct node *top2)
{
    while (top1 != NULL) {
        printf("%d\n", top1->data);
        top1 = top1->next;
```

```
    }
    while (top2 != NULL) {
        printf("%d\n", top2->data);
        top2 = top2->next;
    }
}
int main()
{
    struct queue *q = (struct queue*)malloc(sizeof(struct queue));
    int f = 0, a;
    char ch = 'y';
    q->stack1 = NULL;
    q->stack2 = NULL;
    while (ch == 'y'||ch == 'Y') {
        printf("enter ur choice\n1.add to queue\n2.remove
              from queue\n3.display\n4.exit\n");
        scanf("%d", &f);
        switch(f) {
            case 1 : printf("enter the element to be added to queue\n");
                     scanf("%d", &a);
                     enqueue(q, a);
                     break;
            case 2 : dequeue(q);
                     break;
            case 3 : display(q->stack1, q->stack2);
                     break;
            case 4 : exit(1);
                     break;
            default : printf("invalid\n");
                      break;
        }
    }
```

```
}
```

PROGRAM 6

```
#include<stdlib.h>
#include<stdio.h>

struct bin_tree {
int data;
struct bin_tree * right, * left;
};
typedef struct bin_tree node;

void insert(node ** tree, int val)
{
    node *temp = NULL;
    if(!(*tree))
    {
        temp = (node *)malloc(sizeof(node));
        temp->left = temp->right = NULL;
        temp->data = val;
        *tree = temp;
        return;
    }

    if(val < (*tree)->data)
    {
        insert(&(*tree)->left, val);
    }
    else if(val > (*tree)->data)
    {
        insert(&(*tree)->right, val);
    }
```

```
}



void deltree(node * tree)
{
    if (tree)
    {
        deltree(tree->left);
        deltree(tree->right);
        free(tree);
    }
}

node* search(node ** tree, int val)
{
    if(!(*tree))
    {
        return NULL;
    }

    if(val < (*tree)->data)
    {
        search(&((*tree)->left), val);
    }
    else if(val > (*tree)->data)
    {
        search(&((*tree)->right), val);
    }
    else if(val == (*tree)->data)
    {
```

```
        return *tree;
    }
}

void main()
{
    node *root;
    node *tmp;
      int i;

    root = NULL;
    insert(&root, 2);
    insert(&root, 41);
    insert(&root, 9);
    insert(&root, 18);
    insert(&root, 6);
    insert(&root, 7);
    insert(&root, 14);


    tmp = search(&root, 4);
    if (tmp)
    {
        printf("Searched node=%d\n", tmp->data);
    }
    else
    {
        printf("Data Not found in tree.\n");
    }

    deltree(root);
}
```