

1. Given {4,7,3,2,1,7,9,0} find the location of 7 using Linear and Binary search and also display its first occurrence

```
//Given {4,7,3,2,1,7,9,0} find the location of 7 using  
// Linear search and Binary search and also display its  
// first occurrence
```

```
# include<stdio.h>  
# include<conio.h>  
  
void LINEAR_SEARCH(int a[10], int n, int key)  
{  
    int i,found=0;  
    for(i=0;i<n;i++)  
    {  
        if(key==a[i])  
        {  
            printf("\n %d found at a[%d]",key,i);  
            found=1;  
            break;  
        }  
    }  
    if(found==0)  
        printf("\n Element not found in the list");  
}  
  
void BINARY_SEARCH(int a[10], int key, int first, int last)  
{  
    int mid;  
    mid=(first+last)/2;  
  
    if(key < a[mid])  
        BINARY_SEARCH(a,key,first,mid-1);  
    else if(key > a[mid])  
        BINARY_SEARCH(a,key,mid+1,last);  
    else  
        printf("\n %d found at a[%d]",key,mid);  
}  
  
void DISPLAY_ARRAY(int a[10],int n)  
{  
    int i;  
    printf("\n Given array : ");  
    for(i=0;i<n;i++)  
    {  
        printf("%3d",a[i]);  
    }  
}
```

```

void main()
{
    int a[8] = {4,7,3,2,1,7,9,0};
    int sa[8] = {0,1,2,3,4,7,7,9};
    int n=8;
    int key=7;
    int choice;
    clrscr();
    printf("\n 1. Linear Search");
    printf("\n 2. Binary Search");
    printf("\n Enter your choice :");
    scanf("%d",&choice);
    switch(choice)
    {
        case 1 : {
            DISPLAY_ARRAY(a,n);
            LINEAR_SEARCH(a,n,key);
            break;
        }

        case 2 : {
            DISPLAY_ARRAY(sa,n);
            BINARY_SEARCH(sa,key,0,n-1);
            break;
        }

        default : printf("\n Invalid choice ");
    }
    getch();
}

```

OUTPUT :

LINEAR SEARCH :

```

1. Linear Search
2. Binary Search
Enter your choice :1

Given array : 4 7 3 2 1 7 9 0
7 found at a[1]

```

BINARY SEARCH

```

1. Linear Search
2. Binary Search
Enter your choice :2

Given array : 0 1 2 3 4 7 7 9
7 found at a[5]

```

2. Given {5,3,1,6,0,2,4} order the numbers in ascending order using Bubble Sort Algorithm

// C program to perform Bubble Sort

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

```
# include<conlo.h>
```

```
void BUBBLE_SORT(int a[], int n)
```

```
{
```

```
    int pass,temp,j,l;
```

```
    for(pass=1;pass<=n-1;pass++)
```

```
{
```

```
        for(j=0;j<=n-pass-1;j++)
```

```
{
```

```
            if(a[j] > a[j+1])
```

```
{
```

```
                temp=a[j];
```

```
                a[j]=a[j+1];
```

```
                a[j+1]=temp;
```

```
}
```

```
}
```

```
    printf("\n Array after %d pass --->",pass);
```

```
    for(i=0;i<n;i++)
```

```
        printf("%3d",a[i]);
```

```
}
```

```
}
```

```
void main()
```

```
{
```

```
    int a[7]={5,3,1,6,0,2,4};
```

```
    int n=7;
```

```
    clrscr();
```

```
    printf("\n Input arrays : 5 3 1 6 0 2 4");
```

```
    BUBBLE_SORT(a,n);
```

```
    getch();
```

```
}
```

OUTPUT :

```
Input arrays : 5 3 1 6 0 2 4
Array after 1 pass ---> 3 1 5 0 2 4 6
Array after 2 pass ---> 1 3 0 2 4 5 6
Array after 3 pass ---> 1 0 2 3 4 5 6
Array after 4 pass ---> 0 1 2 3 4 5 6
Array after 5 pass ---> 0 1 2 3 4 5 6
Array after 6 pass ---> 0 1 2 3 4 5 6
```

3(a). Perform the Insertion sort on the input {75,8,1,16,48,3,7,0} and display the output in descending order.

```
// C program to perform Insertion sort
# include<stdio.h>
# include<conio.h>
// Insertion sort

void INSERTION_SORT(int a[], int n)
{
    int pass,k,temp,i,j;
    for(pass=1;pass<n;pass++)
    {
        k=a[pass];
        for(j=pass-1;j>=0 && k<a[j];j--)
        {
            a[j+1] = a[j];
        }
        a[j+1]=k;
        printf("\n\n Sorted arrays after %d pass -->",pass);
        for(i=0;i<n;i++)
        printf("%3d",a[i]);
    }

    printf("\n\n Sorted arrays using Insertion sort in Descending Order\n");
    for(i=n-1;i>=0;i--)
    {
        printf("%3d",a[i]);
    }
}

void main()
{
    int a[8]={75,8,1,16,48,3,7,0};
    int n=8;
    clrscr();
    printf("\n Input arrays : 75,8,1,16,48,3,7,0");
    INSERTION_SORT(a,n);
    getch();
}
```

OUTPUT : INSERTION SORT

Input arrays : 75,8,1,16,48,3,7,0
Sorted arrays after 1 pass --> 8 75 1 16 48 3 7 0
Sorted arrays after 2 pass --> 1 8 75 16 48 3 7 0
Sorted arrays after 3 pass --> 1 8 16 75 48 3 7 0
Sorted arrays after 4 pass --> 1 8 16 48 75 3 7 0
Sorted arrays after 5 pass --> 1 3 8 16 48 75 7 0
Sorted arrays after 6 pass --> 1 3 7 8 16 48 75 0
Sorted arrays after 7 pass --> 0 1 3 7 8 16 48 75
Sorted arrays using Insertion sort in Descending Order
75 48 16 0 7 3 1 0

3(b). Perform the Selection sort on the input {75,8,1,16,48,3,7,0} and display the output in descending order.

```
// C program to perform Selection sort
# include<stdio.h>
# include<conio.h>
// Selection sort

int MIN(int a[], int k, int n)
{
    int loc,j,min;
    min=a[k];
    loc=k;
    for(j=k+1;j<=n-1;j++)
    {
        if(min>a[j])
        {
            min=a[j];
            loc=j;
        }
    }
    return(loc);
}

void SELECTION_SORT(int a[], int k, int n)
{
    int i,loc,temp;
    for(k=0;k<n;k++)
    {
        loc=MIN(a,k,n);
        temp=a[k];
        a[k]=a[loc];
        a[loc]=temp;
        printf("\n\nSorted arrays after %d pass:-->",k);
        for(i=0;i<n;i++)
        printf("%3d",a[i]);
    }
    printf("\n\n Sorted arrays using Selection sort in Descending Order\n");
    for(i=n-1;i>=0;i--)
    {
        printf("%3d",a[i]);
    }
}
void main()
{
    int a[8]={75,8,1,16,48,3,7,0};
    int n=8;
    clrscr();
    printf("\n Input arrays : 75,8,1,16,48,3,7,0");
    SELECTION_SORT(a,0,n);
    getch();
}
```

Input arrays : 75,8,1,16,48,3,7,0
Sorted arrays after 0 pass:--> 0 8 1 16 48 3 7 75
Sorted arrays after 1 pass:--> 0 1 8 16 48 3 7 75
Sorted arrays after 2 pass:--> 0 1 3 16 48 8 7 75
Sorted arrays after 3 pass:--> 0 1 3 7 48 8 16 75
Sorted arrays after 4 pass:--> 0 1 3 7 8 48 16 75
Sorted arrays after 5 pass:--> 0 1 3 7 8 16 48 75
Sorted arrays after 6 pass:--> 0 1 3 7 8 16 48 75
Sorted arrays after 7 pass:--> 0 1 3 7 8 16 48 75
Sorted arrays using Selection sort in Descending Order 75 48 16 8 7 3 1 0

4. Write a program to insert the elements {61,16,8,27} into singly linked list and delete 8,61,27 from the list.
Display your list after each insertion and deletion.

```
// C program to insert {61,16,8,27} and delete {8,61,27} from the list
# include<stdio.h>
# include<conio.h>
# include<alloc.h>
# include<ctype.h>

typedef struct node
{
    int info;
    struct node *link;
}NODE;

NODE *header=NULL;
void DISPLAY()
{
    NODE *start=header;
    printf("\n *** LIST *** : ");
    while(start!=NULL)
    {
        printf("%4d",start->info);
        start=start->link;
    }
}

void INSERT(int item)
{
    NODE *newnode,*curptr;
    newnode = (NODE *) malloc(sizeof(NODE));
    newnode->info=item;
    newnode->link=NULL;

    if(header==NULL)
        header=newnode;
    else
    {
        curptr=header;
        while(curptr->link !=NULL)
        {
            curptr=curptr->link;
        }
        curptr->link=newnode;
    }
    DISPLAY();
}
void DELETE(int item)
{
```

```

NODE *curptr=header, *prevptr=header;

if(header==NULL)
{
    printf("\n EMPTY LIST");
}
else if(header->info==item)
{
    header=header->link;
    free(curptr);
}
else
{
    while(curptr!=NULL)
    {
        if(curptr->info==item)
        {
            prevptr->link=curptr->link;
            free(curptr);
            curptr=curptr->link->link;
        }
        else
        {
            prevptr=curptr;
            curptr=curptr->link;
        }
    }
}
DISPLAY();
}

void main()
{
int item,choice;
clrscr();
printf("\n Insertion :");
INSERT(61);
INSERT(16);
INSERT(8);
INSERT(27);
printf("\n Deletion :");
DELETE(8);
DELETE(61);
DELETE(27);
getch();
}
}

```

Insertion : *** LIST *** : 61 *** LIST *** : 61 16 *** LIST *** : 61 16 8 *** LIST *** : 61 16 8 27	Deletion : *** LIST *** : 61 16 27 *** LIST *** : 16 27 *** LIST *** : 16_
--	--

5. Write a program to Insert the elements {61,16,8,27} into linear queue and delete three elements from the list. Display your list after each insertion and deletion.

```
// Program to Insert the elements {61,16,8,27} into linear queue
// delete three elements from the list
#include <stdio.h>
#define MAX 50

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

void display()
{
    int i;
    if (front == -1)
        printf("\nQueue is empty \n");
    else
    {
        printf("\nQueue is : \n");
        for (i = front; i <= rear; i++)
            printf("%d ", queue_array[i]);
    }
    getch();
}

void insert_Q(int item)
{
    if (rear == MAX - 1)
        printf("Queue Overflow \n");
    else
    {
        if (front == -1)
            front = 0;
        rear = rear + 1;
        queue_array[rear] = item;
    }
    display();
}

void delete_Q()
{
    if (front == -1 || front > rear)
    {
        printf("\nQueue Underflow ");
        return;
    }
```

```

else if(front <= rear)
{
    printf("\nElement deleted from queue is : %d", queue_array[front]);
    front = front + 1;
}
display();

}

void main()
{
    int choice;
    clrscr();
    insert_Q(61);
    insert_Q(16);
    insert_Q(8);
    insert_Q(27);

    delete_Q();
    delete_Q();
    delete_Q();
    getch();
}

```

```

Queue is :
61
Queue is :
61 16
Queue is :
61 16 8
Queue is :
61 16 8 27
Element deleted from queue is : 61
Queue is :
16 8 27
Element deleted from queue is : 16
Queue is :
8 27
Element deleted from queue is : 8
Queue is :
27 -

```

6. Write a program to insert the elements {61,16,8,27} into circular queue and delete 4 elements from the list.
Display your list after each insertion and deletion

```
/*static circular queue*/
#include <stdio.h>
#define size 4
int front = -1;
int rear = -1;
int queue[size];
void display_CQ()
{
    int i;
    printf("\n Circular Queue : ");
    if (front > rear)
    {
        for (i = front; i < size; i++)
        {
            printf("%d ->", queue[i]);
        }
        for (i = 0; i <= rear; i++)
            printf("%d -> ", queue[i]);
    }
    else
    {
        for (i = front; i <= rear; i++)
            printf("%d ->", queue[i]);
    }
    printf("[%d]", queue[front]);
    getch();
}

void insert_CQ(int item)
{
    if ((front == 0 && rear == size - 1) || (front == rear + 1))
    {
        printf("queue is full");
        return;
    }
    else if (rear == -1)
    {
        rear++;
        front++;
    }
    else if (rear == size - 1 && front > 0)
    {
        rear = 0;
    }
    else
    {
        rear++;
    }
}
```

```

        queue[rear] = item;
        display_CQ();
    }

    void delete_CQ()
    {
        if (front == -1)
        {
            printf("Queue is empty ");
        }
        else if (front == rear)
        {
            front = -1;
            rear = -1;
        }
        else
        {
            front++;
        }
        display_CQ();
    }

    int main()
    {
        clrscr();
        printf("\n *** Insertion *** :");
        insert_CQ(61);
        insert_CQ(16);
        insert_CQ(8);
        insert_CQ(27);
        printf("\n *** Deletion *** :");
        delete_CQ();
        delete_CQ();
        delete_CQ();
        delete_CQ();
    }
}

```

OUTPUT :

```

*** Insertion *** :
Circular Queue : 61 ->[61]
Circular Queue : 61 ->16 ->[61]
Circular Queue : 61 ->16 ->8 ->[61]
Circular Queue : 61 ->16 ->8 ->27 ->[61]
*** Deletion *** :
Circular Queue : 16 ->8 ->27 ->[16]
Circular Queue : 8 ->27 ->[8]
Circular Queue : 27 ->[27]
Circular Queue : 0 ->[0]

```

7. Write a program to Insert the elements {61,16,8,27} into ordered singly linked list and delete 8,61,27 from the list. Display your list after each insertion and deletion

```
// C program to Insert {61,16,8,27} In the ordered singly linked list
// and delete 8,61,27 from the list.

#include<stdio.h>
#include<conio.h>

typedef struct node
{
    int info;
    struct node *link;
}NODE;

NODE *header;

void CREATE_HEADER()
{
    header = (NODE *) malloc(sizeof(NODE));
    header->info=0;
    header->link=NULL;
}

void INSERT_ORDERLIST(int item)
{
    NODE *NEWNODE, *PREVPTR, *CURPTR;

    NEWNODE = (NODE *) malloc(sizeof(NODE));
    NEWNODE->info = item;
    NEWNODE->link = NULL;

    if(header->link==NULL)
    {
        header->link=NEWNODE;
    }
    else if(item < header->info)
    {
        NEWNODE->link=header;
        header=NEWNODE;
    }
    else
    {
        PREVPTR=header;
        CURPTR=header->link;
        while(CURPTR!=NULL && item > CURPTR->info)
        {
            PREVPTR=CURPTR;
            CURPTR=CURPTR->link;
        }
        PREVPTR->link=NEWNODE;
        NEWNODE->link=CURPTR;
    }
}
```

```

        }
        PREVPTR->link=NEWNODE;
        NEWNODE->link=CURPTR;
    }

}

void DISPLAY_NODE()
{
    NODE *CURPTR;
    CURPTR=header->link;
    printf("\n LIST : ");
    while(CURPTR!=NULL)
    {
        printf("%d-",CURPTR->info);
        CURPTR=CURPTR->link;
    }
}

void DELETE_NODE(int item)
{
    NODE *PREVPTR, *CURPTR;
    PREVPTR=header;
    CURPTR=header->link;
    if(item == header->info)
    {
        header=CURPTR;
        free(PREVPTR);
    }
    else
    {
        while(CURPTR!=NULL && CURPTR->info !=item)
        {
            PREVPTR=CURPTR;
            CURPTR=CURPTR->link;
        }
        if(CURPTR!=NULL)
        {
            PREVPTR->link=CURPTR->link;
            free(CURPTR);
        }
        else
            printf("\n Data not found");
    }
}

void main()
{
    clrscr();
    CREATE_HEADER();
}

```

```

printf("\n *** INSERTING 61,16,8,27 : ***");
INSERT_ORDERLIST(61);
DISPLAY_NODE();
INSERT_ORDERLIST(16);
DISPLAY_NODE();
INSERT_ORDERLIST(8);
DISPLAY_NODE();
INSERT_ORDERLIST(27);
DISPLAY_NODE();

printf("\n *** DELETE 8,61,27 : ***");
DELETE_NODE(8);
DISPLAY_NODE();
DELETE_NODE(61);
DISPLAY_NODE();
DELETE_NODE(27);
DISPLAY_NODE();
getch();
}

```

```

*** INSERTING 61,16,8,27 : ***
LIST : 61->
LIST : 16->61->
LIST : 8->16->61->
LIST : 8->16->27->61->
*** DELETE 8,61,27 : ***
LIST : 16->27->61->
LIST : 16->27->
LIST : 16->_

```

8. Write a program to add $6x^3 + 10x^2 + 0x + 5$ and $4x^2 + 2x + 1$ using linked list.

```
/*WAP to add Polynomial using linked list*/
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
struct polynomial
{
    int coeff;
    int power;
    struct polynomial *LINK;
};
typedef struct polynomial NODE;
NODE *poly1=NULL,*poly2=NULL,*poly3 = NULL;
NODE *create_poly();
NODE *add_poly(NODE *poly1,NODE *poly2);
void display_poly(NODE *ptr);

/* To create the polynomial*/
NODE *create_poly()
{
    int flag;
    int coeff,pow;
    NODE *tmp_node =(NODE *)malloc(sizeof(NODE)); //create the first node
    NODE *poly=tmp_node;
    do
    {
        printf("\n Enter coeff:");
        scanf("%d",&coeff);
        tmp_node->coeff=coeff;
        printf("\n Enter Pow:");
        scanf("%d",&pow);
        tmp_node->power = pow;
        tmp_node->LINK=NULL;
        printf("\n Do you want to add more terms? (Y=1/N=0):");
        scanf("%d",&flag);
        if(flag==1)
        {
            tmp_node->LINK=(NODE *) malloc(sizeof(NODE));
            tmp_node = tmp_node->LINK;
            tmp_node -> LINK = NULL;
        }
    } while(flag);
    return poly;
}
```

```

/*add two polynomial */
NODE *add_poly(NODE *poly1, NODE *poly2)
{
    NODE *tmp_node,*poly;//Temporary storage for the linked list
    tmp_node=(NODE *)malloc(sizeof(NODE));
    tmp_node->LINK = NULL;
    poly3=tmp_node;
    //Loop while both of the linked list have value
    while(poly1&&poly2)
    {
        if(poly1->power > poly2->power)
        {
            tmp_node->power=poly1->power;
            tmp_node->coeff=poly1->coeff;
            poly1=poly1->LINK;
        }
        else if (poly1->power < poly2->power)
        {
            tmp_node->power = poly2-> power;
            tmp_node->coeff = poly2->coeff;
            poly2 = poly2->LINK;
        }
        else
        {
            tmp_node->power = poly1->power;
            tmp_node->coeff = poly1->coeff+poly2->coeff;
            poly1=poly1->LINK;
            poly2=poly2->LINK;
        }
        if(poly1&&poly2)
        {
            tmp_node->LINK=(NODE *)malloc(sizeof(NODE));
            tmp_node=tmp_node->LINK;
            tmp_node->LINK=NULL;
        }
    }
    //Loop while either of the linked list has value
    while(poly1| poly2)
    {
        tmp_node->LINK =(NODE *)malloc(sizeof(NODE));
        tmp_node=tmp_node->LINK;
        tmp_node->LINK=NULL;
        if(poly1)
        {
            tmp_node->power=poly1->power;
            tmp_node->coeff=poly1->coeff;
            poly1=poly1->LINK;
        }
        if(poly2)
        {
            tmp_node->power = poly2-> power;
            tmp_node->coeff = poly2->coeff;
            poly2 = poly2->LINK;
        }
    }
}

```

```

{
tmp_node->power=poly2->power;
tmp_node->coeff=poly2->coeff;
poly2=poly2->LINK;
}
}
*/
/* Display polynomial */
void display(NODE *ptr)
{
while(ptr!=NULL)
{
printf("%dX^%d",ptr->coeff,ptr->power);
ptr=ptr->LINK;
if(ptr==NULL)
printf(" + ");
}
}
int main()
{
clrscr();
printf("\n Create 1st Polynomial: ");
poly1=create_poly();
printf("\n First polynomial : ");
display(poly1);
printf("\n Create 2nd Polynomial:");
poly2=create_poly();
printf("\n Second polynomial : ");
display(poly2);
add_poly(poly1,poly2);
printf("\n Addition of Two polynomials : ");
display(poly3);
getch();
}
}

```

Create 1st Polynomial:
Enter coeff :6
Enter Pow:3
Do you want to add more terms? (Y=1/N=0):1
Enter coeff :10
Enter Pow:2
Do you want to add more terms? (Y=1/N=0):1
Enter coeff :0
Enter Pow:1
Do you want to add more terms? (Y=1/N=0):1
Enter coeff :5
Enter Pow:0

Do you want to add more terms? (Y=1/N=0):0
First polynomial : $6X^3 + 10X^2 + 6X^1 + 5X^0$
Create 2nd Polynomial:
Enter coeff :4
Enter Pow:2
Do you want to add more terms? (Y=1/N=0):1
Enter coeff :2
Enter Pow:1
Do you want to add more terms? (Y=1/N=0):1
Enter coeff :1
Enter Pow:0
Do you want to add more terms? (Y=1/N=0):0
Second polynomial : $4X^2 + 2X^1 + 1X^0$
Addition of Two polynomials : $6X^3 + 14X^2 + 8X^1 + 6X^0$

9. Write a program to push 5,9,34,17,32 into stack and pop 3 times from the stack, also display the popped numbers

```
//C program to push 5,9,34,17,32 into stack and  
//pop 3 times from the stack (Array implementation)  
  
# include<stdio.h>  
# include<conio.h>  
# define MAX 5  
  
int STACK[MAX];  
int TOP=-1;  
void DISPLAY();  
void PUSH(int item)  
{  
    if(TOP==MAX-1)  
    {  
        printf("\n STACK Overflow");  
    }  
    else  
    {  
        printf("\n ** PUSH %d **",item);  
        TOP=TOP+1;  
        STACK[TOP]=item;  
        DISPLAY();  
    }  
}  
void POP()  
{  
    if(TOP==-1)  
    {  
        printf("\n STACK Underflow");  
        getch();  
    }  
    else  
    {  
        printf("\n ** %d POPED **",STACK[TOP]);  
        TOP=TOP-1;  
        DISPLAY();  
    }  
}  
  
void DISPLAY()  
{  
    int i;  
    for(i=TOP;i>=0;i--)  
    {  
        printf("\n STACK[%d]=%d",i,STACK[i]);  
    }  
}
```

```

    }
    getch();
}
void main()
{
    int i;
    clrscr();
    printf("\n PUSH 5,9,34,17,32\n");
    PUSH(5);
    PUSH(9);
    PUSH(34);
    PUSH(17);
    PUSH(32);
    printf("\n POP 3 elements\n");
    POP();
    POP();
    POP();
}

```

PUSH 5,9,34,17,32

** PUSH 5 **
 STACK[0]=5
 ** PUSH 9 **
 STACK[1]=9
 STACK[0]=5
 ** PUSH 34 **
 STACK[2]=34
 STACK[1]=9
 STACK[0]=5
 ** PUSH 17 **
 STACK[3]=17
 STACK[2]=34
 STACK[1]=9
 STACK[0]=5
 ** PUSH 32 **
 STACK[4]=32
 STACK[3]=17
 STACK[2]=34
 STACK[1]=9
 STACK[0]=5_

** 32 POPED **
 STACK[3]=17
 STACK[2]=34
 STACK[1]=9
 STACK[0]=5
 ** 17 POPED **
 STACK[2]=34
 STACK[1]=9
 STACK[0]=5
 ** 34 POPED **
 STACK[1]=9
 STACK[0]=5_

10. Write a recursive program to find GCD of 4,6,8.

// C program to find GCD (4,6,8) using recursion.

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

int GCD(int m, int n)
{
    if(n==0)
    {
        return(m);
    }
    else if(n>m)
    {
        return(GCD(n,m));
    }
    else
    {
        return(GCD(n,m%n));
    }
}

void main()
{
    // three nos 4,6,8
    int gcd12, gcd3;
    clrscr();
    gcd12=GCD(4,6);
    printf("\n GCD between 4 & 6 = %d",gcd12);
    gcd3=(GCD(gcd12,8));
    printf("\n GCD between 4,6 & 8 = %d",gcd3);
    getch();
}
```

```
GCD between 4 & 6 = 2
GCD between 4,6 & 8 = 2
```

12. Write a program to convert an infix expression $x^y/(5*z)+2$ to its postfix expression

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

#define SIZE 50

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

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

char pop()
{
    return(stack[top--]);
}

int pr(char symbol)
{
    if(symbol == '^')
    {
        return(3);
    }
    else if(symbol == '*' || symbol == '/')
    {
        return(2);
    }
    else if(symbol == '+' || symbol == '-')
    {
        return(1);
    }
    else
    {
        return(0);
    }
}

void main()
{
    char infix[50],postfix[50],ch,elem;
    int i=0,k=0;
    clrscr();
    printf("Enter Infix Expression : ");
    scanf("%s",infix);
    push('#');
    while( (ch=infix[i++]) != '\0')
```

```

{
    if( ch == '(') push(ch);
    else
        if(isalnum(ch)) postfix[k++]=ch;
        else
            if( ch == ')')
            {
                while( stack[top] != '(')
                    postfix[k++]=pop();
                elem=pop();
            }
            else
            {
                while( pr(stack[top]) >= pr(ch) )
                    postfix[k++]=pop();
                push(ch);
            }
    }
    while( stack[top] != '#' )
        postfix[k++]=pop();
    postfix[k]='\0';
    printf("\nPostfix Expression = %s\n",postfix);
    getch();
}

```

Enter Infix Expression : $x^y/(5*z)+2$

Postfix Expression = xy^5z*/2+

13. Write a program to evaluate a postfix expression $5\ 3+8\ 2-\ *$.

```
#include<stdio.h>
int stack[20];
int top = -1;

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

int pop()
{
    return stack[top--];
}

int main()
{
    char *postfix;
    int A,B,RES,num;
    clrscr();
    printf("Enter the expression :: ");
    scanf("%s",postfix);
    while(*postfix != '\0')
    {
        if(isdigit(*postfix))
        {
            num = *postfix - 48; // converting char into num
            push(num);
        }
        else
        {
            A = pop();
            B = pop();
            switch(*postfix)
            {
                case '+': RES = B + A; break;
                case '-': RES = B - A; break;
                case '*': RES = B * A; break;
                case '/': RES = B / A; break;
            }
            push(RES);
        }
        postfix++;
    }
    printf("\nThe result of expression = %d\n\n",pop());
    getch();
}
```

Enter the expression :: 53+82-*

The result of expression = 48

15. Write a program to create binary search tree with the elements {2,5,1,3,9,0,6} and perform inorder, preorder and post order traversal.

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

typedef struct BST {
    int data;
    struct BST *lchild, *rchild;
} node;

node *create_node() {
    node *temp;
    temp = (node *) malloc(sizeof(node));
    temp->lchild = NULL;
    temp->rchild = NULL;
    return temp;
}

void insert(node *root, node *new_node) {
    if (new_node->data < root->data) {
        if (root->lchild == NULL)
            root->lchild = new_node;
        else
            insert(root->lchild, new_node);
    }
    if (new_node->data > root->data) {
        if (root->rchild == NULL)
            root->rchild = new_node;
        else
            insert(root->rchild, new_node);
    }
}

void inorder(node *temp) {
    if (temp != NULL) {
        inorder(temp->lchild);
        printf("%3d", temp->data);
        inorder(temp->rchild);
    }
}

void preorder(node *temp) {
    if (temp != NULL) {
        printf("%3d", temp->data);
        preorder(temp->lchild);
        preorder(temp->rchild);
    }
}

void postorder(node *temp) {
    if (temp != NULL) {
```

```

postorder(temp->lchild);
postorder(temp->rchild);
printf("%3d", temp->data);
}
}
void main()
{
int n=7,i=1;
node *new_node, *root;
node *create_node();
root = NULL;
clrscr();

printf("\nProgram For Binary Search Tree ");
for(i=1;i<=n;i++)
{
    new_node = create_node();
    printf("\nEnter The Element ");
    scanf("%d", &new_node->data);

    if (root == NULL) /* Tree is not Created */
        root = new_node;
    else
        insert(root, new_node);
}
printf("\nThe Inorder display : ");
inorder(root);
printf("\nThe Preorder display : ");
preorder(root);
printf("\nThe Postorder display : ");
postorder(root);
getch();
}

```

Program For Binary Search Tree

Enter The Element 2

Enter The Element 5

Enter The Element 1

Enter The Element 3

Enter The Element 9

Enter The Element 0

Enter The Element 6

The Inorder display : 0 1 2 3 5 6 9

The Preorder display : 2 1 0 5 3 9 6

The Postorder display : 0 1 3 6 9 5 2

16. Write a program to Sort the following elements using heap sort {9,16,32,8,4,1,5,8,0}

```
// Heap Sort
# include<stdio.h>
void heapify(int a[], int n, int i)
{
    int largest = i;
    int left = 2 * i + 1;
    int right = 2 * i + 2;

    if(left < n && a[left] > a[largest])
    {
        largest = left;
    }

    if(right < n && a[right] > a[largest])
    {
        largest = right;
    }
    if(largest != i)
    {
        int temp;
        temp = a[i];
        a[i] = a[largest];
        a[largest] = temp;
        heapify(a,n,largest);
    }
}

void HEAPSORT(int a[], int n)
{
    int i;
    for(i=n/2-1; i>=0; i--)
    heapify(a,n,i);
    for( i=n-1; i>=0; i--)
    {
        int temp;
        temp = a[0];
        a[0] = a[i];
        a[i] = temp;
        heapify(a,i,0);
    }
}
void printArr(int arr[], int n)
{
    int i;
    for( i=0; i<n; ++i)
    {
        printf("%4d",arr[i]);
    }
}
```

```
}

void main()
{
    int a[] = { 9,16,32,8,4,1,5,8,0 };
    int n = sizeof(a) / sizeof(a[0]);
    clrscr();
    printf("\n Before sorting : ");
    printArr(a,n);
    HEAPSORT(a,n);
    printf("\n After sorting : ");
    printArr(a,n);
    getch();
}
```

```
Before sorting : 9 16 32 8 4 1 5 8 0
After sorting : 0 1 4 5 8 0 9 16 32
```

17. Given S1={"Flowers"} ; S2={"are beautiful"} I. Find the length of S1 II. Concatenate S1 and S2 III. Extract the substring "low" from S1 IV. Find "are" in S2 and replace it with "is"

```
// S1 = {"Flowers"} S2={"are beautiful"}  
// Find : (a) Length of S1 (b) Concatenate S1 and S2  
// (c) Extract the substring "low" from S1  
// (d) Replace "are" with "is" in S2  
  
# include<stdio.h>  
# include<conio.h>  
# include<string.h>  
  
int LENGTH(char *str)  
{  
    int i=0, len=0;  
    while(str[i]!='\0')  
    {  
        len++;  
        i++;  
    }  
    return(len);  
}  
  
void CONCAT(char *str1, char *str2)  
{  
    int i=0,j=0;  
    while(str1[i]!='\0')  
    {  
        i++;  
    }  
    while(str2[j]!='\0')  
    {  
        str1[i]=str2[j];  
        i++;  
        j++;  
    }  
    str1[i]='\0';  
    printf("\n Concatenated string = %s",str1);  
}  
void EXTRACT(char *str,int pos, int elen)  
{  
    int i=0,j=0;  
    char substr[10];  
    for(i=pos;i<=elen;i++)  
    {  
        substr[j]=str[i];  
        j++;  
    }  
    substr[j]='\0';  
    printf("\n Substring = %s",substr);  
}
```

```

void REPLACE(char *str, char *sstr, char *rstr, int pos)
{
    char output[50];
    int i=0,j=0,k=0;
    for(i=0;i<LENGTH(str);i++)
    {
        if(i==pos)
        {
            for(k=pos;k<LENGTH(rstr);k++)
            {
                output[j]=rstr[k];
                j++;
                i++;
            }
        }
        else
        {
            output[j]=str[i];
            j++;
        }
    }
    output[j]='\0';
    printf("\n Output = %s",output);
    getch();
}

void main()
{
    char *S1,*S2;
    int len,choice,pos,elen;
    while(1)
    {
        clrscr();
        strcpy(S1,"Flowers");
        strcpy(S2,"are beautiful");
        printf("\n S1 = %s S2 = %s",S1,S2);
        printf("\n 1. Length 2.Concatenate 3.Extract Substring 4.REPLACE 5.Exit\n");
        printf("\n Enter your choice : ");
        scanf("%d",&choice);
        switch(choice)
        {
        case 1 : {
            len = LENGTH(S1);
            printf("\n Length of %s = %d",S1,len);
            }break;
        case 2: {
            CONCAT(S1,S2);
            }break;
        case 3: { printf("\n Enter position & length of substring in S1 : ");
        }
    }
}

```

```

        scanf("%d %d",&pos,&elen);
        EXTRACT(S1,pos,elen);
    }break;

case 4: {
    REPLACE(S2,"are","is",0);
}break;
case 5: exit(0);
default : printf("\n Invalid option");
}
getch();
}
}

```

S1 = Flowers S2 = are beautiful
 1. Length 2.Concatenate 3.Extract Substring 4.REPLACE 5.Exit
 Enter your choice : 1
 Length of Flowers = ?_

S1 = Flowers S2 = are beautiful
 1. Length 2.Concatenate 3.Extract Substring 4.REPLACE 5.Exit
 Enter your choice : 2
 Concatenated string = Flowersare beautiful

S1 = Flowers S2 = are beautiful
 1. Length 2.Concatenate 3.Extract Substring 4.REPLACE 5.Exit
 Enter your choice : 3
 Enter position & length of substring in S1 : 1 3
 Substring = low

S1 = Flowers S2 = are beautiful
 1. Length 2.Concatenate 3.Extract Substring 4.REPLACE 5.Exit
 Enter your choice : 4
 Output = is beautiful_