KLE SOCIETY's

S. NIJALINGAPPA COLLEGE

DEPARTMENT OF COMPUTER SCIENCE

Bachelor of Computer Applications

DATA STRUCTURES LAB Manual

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.

```
#include<stdio.h>
#include<conio.h>
void main()
{
        int flag=1,a[100],ele,n,i,j,lb,ub,mid,ch,temp;
        int c=0;
        clrscr();
        while(c!=3)
                {
                        printf("\n 1 Binary search\n");
                        printf("\n 2.Linear search\n");
                        printf("\n 3.Exit\n");
                        printf("\n enter your choice\n");
                        scanf("%d",&ch);
                        switch(ch)
                                {
                                  case 1:printf("\n binary search\n");
                                      printf("\n enter the size\n");
                                      scanf("%d",&n);
                                      printf("\n enter the elements 1 by 1\n");
                                      for(i=0;i<n;i++)
                                              {
                                                      scanf("%d",&a[i]);
                                              }
                                      printf("\n enter the elements to be searched\n");
                                      scanf("%d",&ele);
```

```
for(i=0;i<n;i++)
        {
                for(j=0;j<n-1;j++)
                        {
                                if(a[j]>a[j+1])
                                        {
                                                temp=a[j];
                                                a[j]=a[j+1];
                                                a[j+1]=temp;
                                        }
                        }
        }
printf("\n array after sorting\n");
for(i=0;i<n;i++)
        {
printf("%d\n",a[i]);
        }
lb=0; ub=n-1;
mid=(lb+ub)/2;
while(a[mid]!=ele && lb<=ub)
        {
                if(ele>a[mid])
                lb=mid+1;
                else ub=mid-1;
                mid=(lb+ub)/2;
         }
if(a[mid]==ele)
printf("\n search element to found at %d\n",mid+1);
else
printf("\n search element is not possible\n");
break;
case 2:printf("\n linear search\n");
printf("\n enter the size\n");
scanf("%d",&n);
printf("\n enter the elements 1 by 1\n");
for(i=0;i<n;i++)
        {
```

```
                    scanf("%d",&a[i]);
                }
            printf("\n enter the elements to be searched\n");
            scanf("%d",&ele);
            for(i=0;i<n;i++)
                {
                    if(a[i]==ele)
                        {
                          printf("%d is availabe at %d\n",ele,i+1);
                           flag=0;
                        }
                }
            if(i==n && flag==1)
                {
                    printf(" %d is not availabel\n",ele);
                }
                break;
            default :printf("\n Exiting the program\n");
                    getch();
                    exit(0);
        }
    }
    getch();
}
```

Output:

```
 1. Binary search
 2. linear search
 Enter your choice :
1
Given Array :
  4  7  3  2  1  7  9  0
Arrange the elements in ascending order
Sorted list in ascending order:
0
1
2
3
4
7
7
9
7 found at location 6
_
```

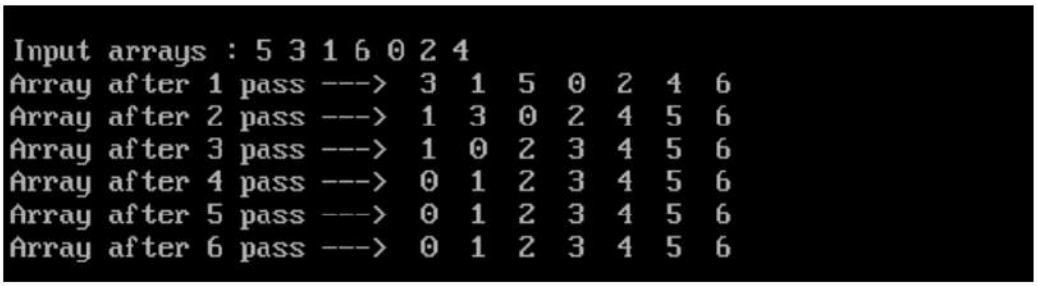
```
 1. Binary search
 2. linear search
 Enter your choice :
2
Given Array :
  4  7  3  2  1  7  9  0
 7 found at 2
```

```
 1. Binary search
 2. linear search
 Enter your choice :
3
 invalid choice
```

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

```
void Bubblesort( int a[], int n)
{
        int  pass,temp,i,j;
        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("%4d",a[i]);
                }
}
void main()
{
        int a[]={5,3,1,6,0,2,4};
        int n=7;
        clrscr();
        printf("\n input arrays:5 3 1 6 0 2 4 ");
        Bubblesort(a,n);
        getch();
}
```

Output:

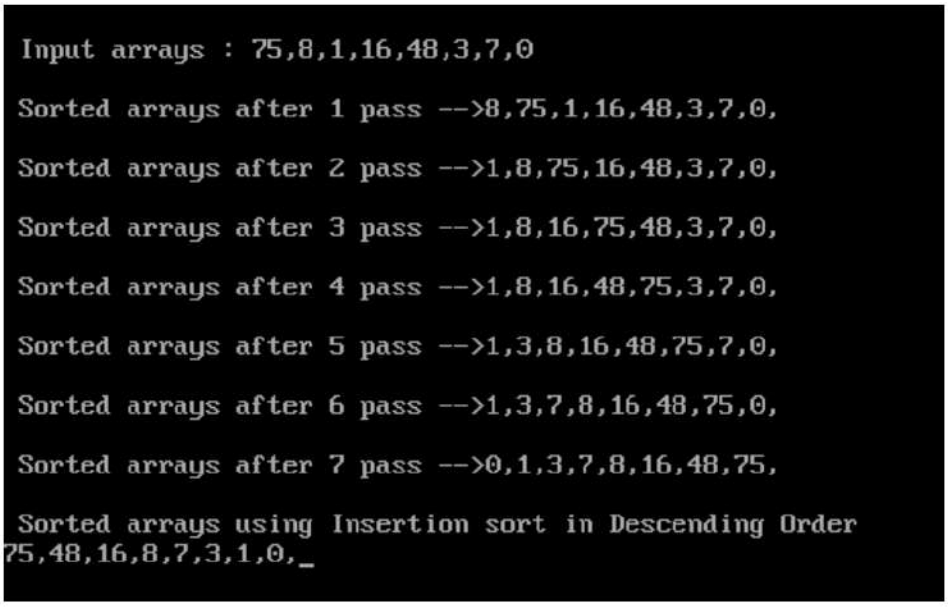

Input arrays : 5 3 1 6 0 2 4
Array after 1 pass ---> 3 1 5 0 2 4 6
Array after 2 pass ---> 1 3 0 2 4 5 6
Array after 3 pass ---> 1 0 2 3 4 5 6
Array after 4 pass ---> 0 1 2 3 4 5 6
Array after 5 pass ---> 0 1 2 3 4 5 6
Array after 6 pass ---> 0 1 2 3 4 5 6

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



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("%d,",a[i]);
                }
       printf("\n\n Sorted arrays using Selection sort in Descending Order\n");
```

```
    for(i=n-1;i>=0;i--)
        {
            printf("%d,",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();
}
```

Output:

```
 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.**

```
# 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();
}
```

Output:

```
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.

```
#define MAX 5
#include<stdio.h>
int front = 0, rear = -1;
int queue[MAX];
main()
{
    void qinsert();
    void qdisplay();
    void qdelete();
    int  choice;
    clrscr();
    while(1)
        {
            printf("\n\n Linear Queue simulator:");
            printf("\n 1. ADD 2. DELETE 3.DISPLAY 4. EXIT");
            printf("\n ENTER YOUR CHOICE: ");
            scanf("%d", &choice);
            switch(choice)
                {
                    case 1: qinsert();
                            qdisplay();
                             break;
                    case 2: qdelete();
                            qdisplay();
                             break;
                    case 3: qdisplay();
                            break;
                    case 4: exit(0);
                    default : printf("\n ERROR IN CHOICE");
                }
```

```
        }
    }
void qinsert()
{
    int item;
    if(rear==MAX-1)
    printf("\n QUEUE IS EMPTY");
    else
        {
             printf("\n Enter an item :");
             scanf("%d",&item);
            rear++;
             queue[rear]=item;
        }
}
void qdelete()
{
    if(rear==front-1)
     printf("\n Queue Underflow");
    else if(rear==front)
       {
            printf("\n this is the last element in the queue ");
            printf("\n The last element deleted is : %d",queue[front]);
             front=0;
            rear=-1;
        }
    else
        {
             printf("\n deleted item is %d", queue[front]);
            front++;
        }
}
```

```
void qdisplay()
{
    int i;
    if (rear==front-1)
     printf("\n No elements in queue");
    else
        {
            printf("\n Queue elements are :");
            for(i=front;i<=rear;i++)
            printf("\t%d\t",queue[i]);
        }
}
```

Output:

```
Linear Queue simulator:
1. ADD 2. DELETE 3.DISPLAY 4. EXIT
ENTER YOUR CHOICE: 1

Enter an item :61

Queue elements are :   61

Linear Queue simulator:
1. ADD 2. DELETE 3.DISPLAY 4. EXIT
ENTER YOUR CHOICE: 1

Enter an item :16

Queue elements are :   61              16

Linear Queue simulator:
1. ADD 2. DELETE 3.DISPLAY 4. EXIT
ENTER YOUR CHOICE: 1

Enter an item :8
```

```
Queue elements are :   61              16              8

Linear Queue simulator:
1. ADD 2. DELETE 3.DISPLAY 4. EXIT
ENTER YOUR CHOICE: 1

Enter an item :27

Queue elements are :   61              16              8               27


Linear Queue simulator:
1. ADD 2. DELETE 3.DISPLAY 4. EXIT
ENTER YOUR CHOICE: 3

Queue elements are :   61              16              8               27


Linear Queue simulator:
1. ADD 2. DELETE 3.DISPLAY 4. EXIT
ENTER YOUR CHOICE: 2_
```

```
deleted item is 61
Queue elements are :   16              8               27

Linear Queue simulator:
1. ADD 2. DELETE 3.DISPLAY 4. EXIT
ENTER YOUR CHOICE: 2

deleted item is 16
Queue elements are :   8               27

Linear Queue simulator:
1. ADD 2. DELETE 3.DISPLAY 4. EXIT
ENTER YOUR CHOICE: 2

deleted item is 8
Queue elements are :   27

Linear Queue simulator:
1. ADD 2. DELETE 3.DISPLAY 4. EXIT
ENTER YOUR CHOICE: 4
```

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.**

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
#include<malloc.h>
void CQdisplay();
void CQdelete();
void CQinsert(int item);

struct queue
        {
                int info;
                struct queue *link;
        };
struct queue *front=NULL,*rear=NULL;
void CQinsert(int item)
  {
        struct queue *newnode;
        newnode=(struct queue*)malloc(sizeof(struct queue));
        newnode->info=item;
        newnode->link=NULL;
        if(front==NULL && rear==NULL)
                {
                        front=rear=newnode;
                        rear->link=front;
                }
        else
                {
                        rear->link=newnode;
                        rear=newnode;
                        rear->link=front;
                }
   }
```

```
void CQdelete()
  {
       struct queue *ptr;
       ptr=front;
       if(front==NULL && rear==NULL)
       printf("\n Queue is empty");
       else if(front==rear)
              {
                     front=rear=NULL;
                     printf("\n the value being deleted is: %d",ptr->info);
                     free(ptr);
              }
              else
              {
                     front=front->link;
                     rear->link=front;
                     printf("\n thevalue being deleted is :%d",ptr->info);
                     free(ptr);
              }
   }
void CQdisplay()
  {
       struct queue *ptr;
       ptr=front;
       if(front==NULL&&rear==NULL)
       printf("\n Queue is empty");
       else
              {
                     printf("\n the queue elements are:");
                     do
                            {
                                   printf("%d\t",ptr->info);
                                   ptr=ptr->link;
                            }while(ptr!=front);
              }
```

```
   }
void main()
{
        int val,choice;
        clrscr();
        do
        {
                printf("\n 1. Insert 2.delete 3.display 4.exit");
                printf("\n Enter your choice: ");
                scanf("%d",&choice);
                switch(choice)
                    {
                case 1: printf(" \n Enter the elemnets to insert into queue:");
                        scanf("%d",&val);
                        CQinsert(val);
                        break;
                case 2: CQdelete();
                          break;
                case 3: CQdisplay();
                          break;
                      }
        }while(choice!=4);
        getch();
}
```

Output:

```
Enter your choice: 1

Enter the elemnets to insert into queue:61

1. Insert 2.delete 3.display 4.exit
Enter your choice: 1

Enter the elemnets to insert into queue:16

1. Insert 2.delete 3.display 4.exit
Enter your choice: 1

Enter the elemnets to insert into queue:8

1. Insert 2.delete 3.display 4.exit
Enter your choice: 1

Enter the elemnets to insert into queue:27

1. Insert 2.delete 3.display 4.exit
Enter your choice: 3

the queue elements are:61      16      8       27
1. Insert 2.delete 3.display 4.exit
Enter your choice: 2
```

```
Enter your choice: 2

thevalue being deleted is :61
1. Insert 2.delete 3.display 4.exit
Enter your choice: 3

the queue elements are:16       8       27
1. Insert 2.delete 3.display 4.exit
Enter your choice: 2

thevalue being deleted is :16
1. Insert 2.delete 3.display 4.exit
Enter your choice: 3

the queue elements are:8        27
1. Insert 2.delete 3.display 4.exit
Enter your choice: 2

thevalue being deleted is :8
1. Insert 2.delete 3.display 4.exit
Enter your choice: 3

the queue elements are:27
1. Insert 2.delete 3.display 4.exit
Enter your choice: 2
```

```
the value being deleted is: 27
1. Insert 2.delete 3.display 4.exit
Enter your choice: 3

Queue is empty
```

```
1. Insert 2.delete 3.display 4.exit
Enter your choice: 4
```

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.

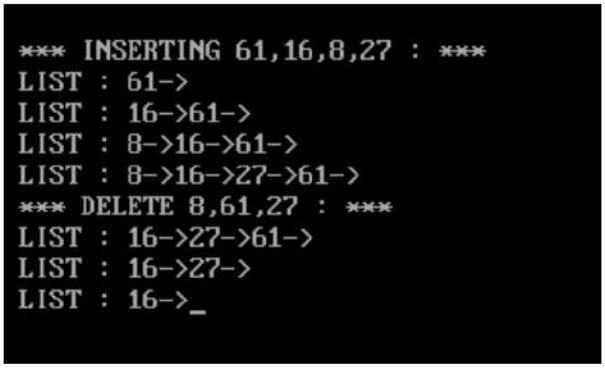
```
# include<stdio.h>
# include<conio.h>
typedef struct node
{
        int info;
        struct node *link;
}NODE;
NODE *header;
void CREATE_HEADER()
{
        header = (NODE *) malloc(sizeof(NODE));
       header->info=0;
       header->link=NULL;
}
void INSERT_ORDERLIST(int item)
   {
       NODE *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;
             }
    }
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();
```

}

Output:



8. Write a program to add 6x3+10x2+0x+5 and 4x2+2x+1 using linked list.

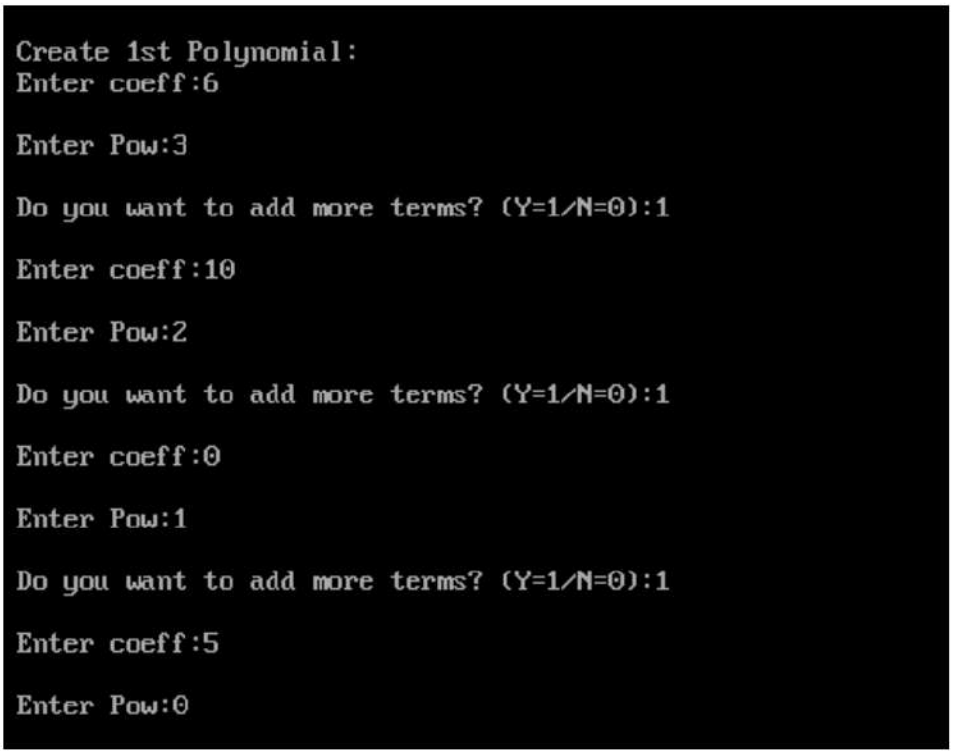
```
/*WAP to add Polynomial using linked list*/
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
struct polynomial
{
        int coeff;
        int power;
       struct polynomial *LINK;
};
typedef struct polynomial NODE;
NODE *poly1=NULL,*poly2=NULL,*poly3 = NULL;
NODE *create_poly();
NODE *add_poly(NODE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

```
            while(ptr!=NULL)
              {
                    printf("%dX^%d",ptr->coeff,ptr->power);
                    ptr=ptr->LINK;
                    if(ptr!=NULL)
                    printf(" + ");
              }
        }
void 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();
}
```

Output:


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 + 0X^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 + 2X^1 + 6X^0
```

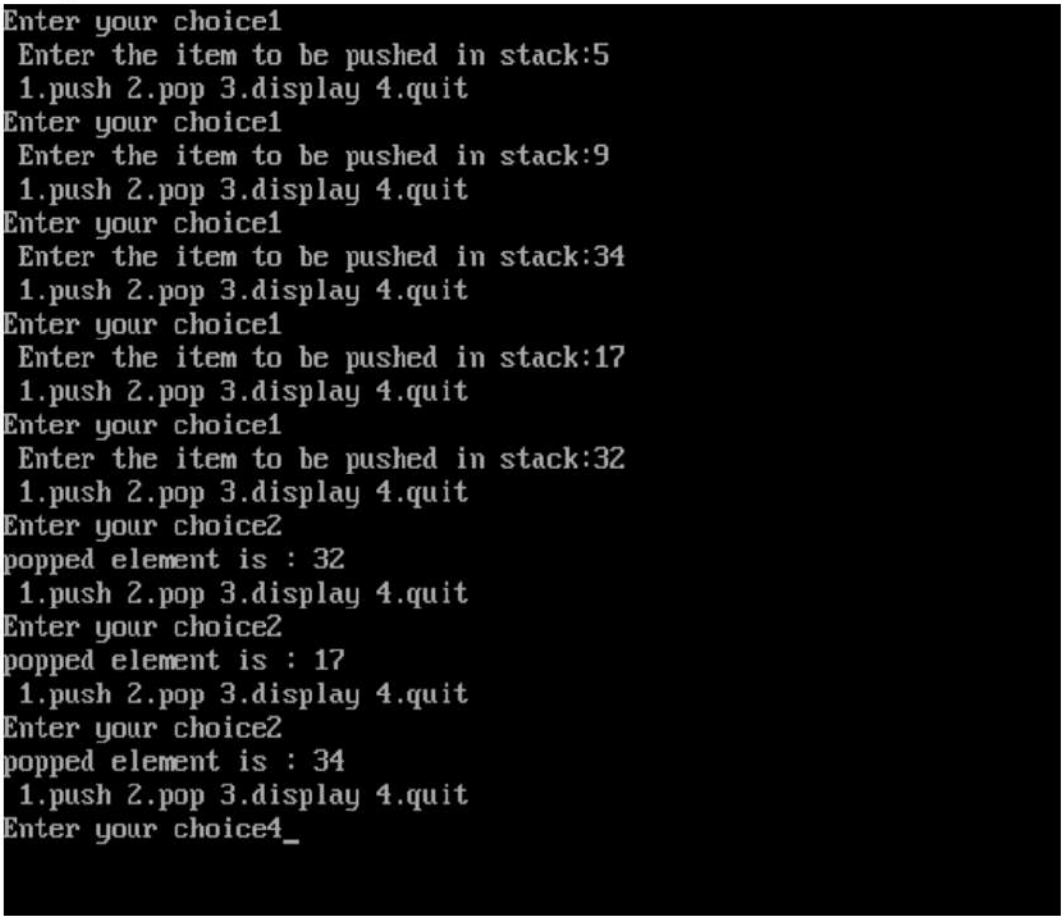
9 . Write a program to push 5, 9,34,17,32 into stack and pop 3 times from the stack, also display the poppednumbers

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#define MAXSTK 5
int TOP=-1;
int s[MAXSTK];
void push();
void pop();
void display();
void main()
{
        int choice;
        clrscr();
        while(1)
                {
                        printf(" 1.push 2.pop 3.display 4.quit\n");
                        printf("Enter your choice");
                        scanf("%d",&choice);
                        switch(choice)
                                {
                                        case 1: push();
                                        break;
                                        case 2: pop();
                                        break;
                                        case 3: display();
                                        break;
                                        case 4: exit(1);
                                        default : printf(" wrong choice \n");
                                }
```

```
            }
        getch();
}
void push()
{
        int item;
        if(TOP==(MAXSTK-1))
        printf(" stack overflow \n");
        else
                {
                        printf(" Enter the item to be pushed in stack:");
                        scanf(" %d",&item);
                        TOP=TOP+1;
                        s[TOP]=item;
                }
}
void pop()
{
        if(TOP==-1)
        printf("stack underflow\n");
        else
                {
                        Printf("popped element is : %d \n",s[TOP]);
                        TOP=TOP-1;
                }
}
void display()
{
        int i;
        if(TOP==-1)
        printf("Stack is empty \n");
        else
                {
                        printf("Stack elements :\n");
                        for(i=TOP;i>=0;i--)
```

```
                printf("%d\n",s[i]);
            }
}
```

Output:

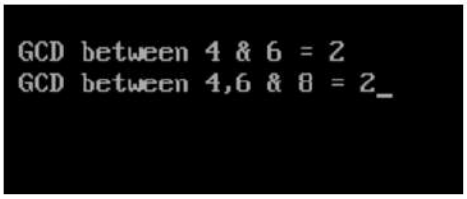


10 . Write a recursive program to find GCD of 4,6,8.

```
// 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()
{
        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();
}
```

Output:



11.Write a program to insert the elements {5,7,0,6,3,9} into circular queue and delete 6,9&5 from it(using linked list implementation)…

```
#include<stdio.h>
#include<conio.h>
#include<malloc.h>
#include<stdlib.h>
struct queue
{
        int info;
        struct queue *link;
};
struct queue *front=NULL,*rear=NULL;
void Qinsert(int item)
{
```

```
        struct queue *new_node;
        new_node=(struct queue*)malloc(sizeof(struct queue));
        new_node->info=item;
        new_node->link=NULL;
        if(front==NULL && rear==NULL)
                {
                        front=rear=new_node;
                        rear->link=front;
                }
        else
                {
                        rear->link=new_node;
                        rear=new_node;
                        rear->link=front;
                }
   }
void Qdelete()
{
        struct queue *ptr;
        ptr=front;
        if(front==NULL && rear==NULL)
        printf("\n Queue is empty");
        else if(front==rear)
                {
                        front=rear=NULL;
                        printf("\n The value being deleted is : %d", ptr->info);
                        free(ptr);
                }
        else
                {
                        front=front->link;
                        rear->link=front;
                        printf("\n the value being deleted is : %d",ptr->info);
                        free(ptr);
                }
```

```
 }
void display()
{
        struct queue *ptr;
        ptr=front;
        if(front==NULL && rear== NULL)
                printf("\n Queue is empty:");
        else
                {
                        printf("\n The queue elements are :");
                        do
                                {
                                        printf("%d\t",ptr->info);
                                        ptr=ptr->link;
                                }while(ptr!=front);
                }
}
void main()
{
        int val, choice;
        clrscr();
        do
                {
                        printf("\n ***** main menu**********");
                        printf("\n 1. insert 2.delete 3.display 4.exit");
                        printf("\n Enter your choice:");
                        scanf("%d",&choice);
                        switch(choice)
                                {
                        case 1: printf("\n Enter the number to insert into queue:");
                                scanf("%d",&val);
                                Qinsert(val);
                                break;
                        case 2: Qdelete();
                                break;
```

```
                case 3: display();
                        break;
                        }
            }while(choice!=4);
            getch();
  }
```

Output:

```
***** main menu*************
1. insert 2.delete 3.display 4.exit
Enter your choice:1

Enter the number to insert into queue:5

***** main menu*************
1. insert 2.delete 3.display 4.exit
Enter your choice:1

Enter the number to insert into queue:7

***** main menu*************
1. insert 2.delete 3.display 4.exit
Enter your choice:1

Enter the number to insert into queue:0

***** main menu*************
1. insert 2.delete 3.display 4.exit
Enter your choice:1

Enter the number to insert into queue:6
```

```
***** main menu************
1. insert 2.delete 3.display 4.exit
Enter your choice:1

Enter the number to insert into queue:3

***** main menu************
1. insert 2.delete 3.display 4.exit
Enter your choice:1

Enter the number to insert into queue:9

***** main menu************
1. insert 2.delete 3.display 4.exit
Enter your choice:3

The queue elements are :5     7      0      6      3      9
***** main menu************
1. insert 2.delete 3.display 4.exit
Enter your choice:2

the value being deleted is : 5
***** main menu************
1. insert 2.delete 3.display 4.exit
Enter your choice:2
```

```
the value being deleted is : 7
****** main menu*************
1. insert 2.delete 3.display 4.exit
Enter your choice:2

the value being deleted is : 0
****** main menu*************
1. insert 2.delete 3.display 4.exit
Enter your choice:2

the value being deleted is : 6
****** main menu*************
1. insert 2.delete 3.display 4.exit
Enter your choice:2

the value being deleted is : 3
****** main menu*************
1. insert 2.delete 3.display 4.exit
Enter your choice:2

The value being deleted is : 9
****** main menu*************
1. insert 2.delete 3.display 4.exit
Enter your choice:3_
```

```
Enter your choice:3

Queue is empty:
****** main menu**************
1. insert 2.delete 3.display 4.exit
Enter your choice:4
```

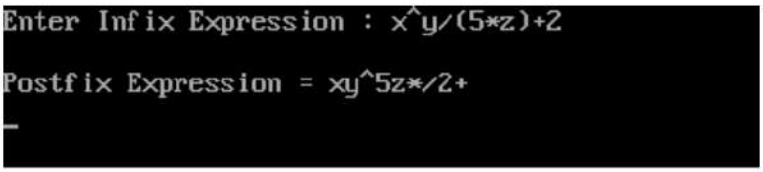
12. Write a program to convert an infix expression x^y/(5*z)+2 to its postfix expression.

```
#include <stdio.h>
#include <ctype.h>
#define SIZE 50
char stack[SIZE];
int top=-1;
int push(char elem)
 {
        stack[++top]=elem;
        return 0;
 }
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();
}
```

Output:

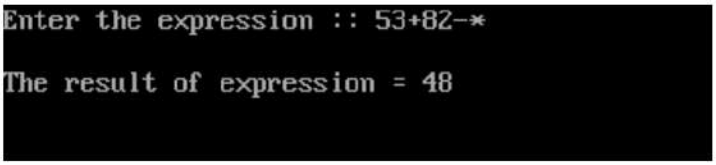


13. Write a program to evaluate a postfix expression 5 3+8 2 - *.

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

Output:



14. Write a program to create a binary tree with the elements {18,15,40,50,30,17,41} after creation insert 45 and 19 into tree and delete 15,17 and 41 from tree. Display the tree on each insertion and deletion operation.

```
#include<stdio.h>
#include<stdlib.h>
struct Node
{
int data;
struct Node* left;
struct Node* right;
};

void insertNode(struct Node** subtree,int key)
{
struct Node* temp = NULL;
if(!(*subtree))
{
   temp = (struct Node*)malloc(sizeof(struct Node));
     temp->data = key;
   temp->left = NULL;
   temp->right = NULL;
   *subtree = temp; return;
}
if(key<(*subtree)->data)
  {
          insertNode(&(*subtree)->left,key);
  }

else if(key>(*subtree)->data)
  {
          insertNode(&(*subtree)->right,key);
  }
}

struct Node* minValue(struct Node* node)
```

```
{
struct Node* temp = node;
while(temp->left != NULL)
{
temp = temp->left;
}
return temp;
}

struct Node* deleteNode(struct Node* root,int key)
{
if(root == NULL)
  {
          return root;
  }
if(key < root->data)
  {
          root->left = deleteNode(root->left,key);
  }
else if(key > root->data)
  {
          root->right = deleteNode(root->right,key);
  }
else
  {
          if(root->left == NULL)
                  {
                          struct Node* temp = root->right; free(root);
                          return temp;
                  }
          else if(root->right == NULL)
                  {
                          struct Node* temp = root->left;
                          free(root);
                          return temp;
                  }
                  {
```

```
                        struct Node* temp = minValue(root->right);
                        root->data = temp->data;
                        root->right = deleteNode(root->right,temp->data);
                    }
  }
return root;
}
void preorder(struct Node* tree)
{ //root,left,right
  if(tree)
    {
        printf("%d ",tree->data);
        preorder(tree->left);
        preorder(tree->right);
     }
}

int main()
{
  struct Node* root = NULL;
  clrscr();
  insertNode(&root,18);
  insertNode(&root,15);
  insertNode(&root,40);
  insertNode(&root,50);
  insertNode(&root,30);
  insertNode(&root,17);
  insertNode(&root,41);
  printf("\n");
  printf("Nodes  of the tree are: ");
  preorder(root);
  printf("\n After inserting 45 and 19 nodes in tree :");
  insertNode(&root,45);
  insertNode(&root,19);
  printf("\n\n");
  preorder(root);
  printf("\n\n");
```

```
    deleteNode(root,15);
    printf("\n After deleting 15 from tree :");
    printf("\n nodes in the tree :");
    preorder(root);
    printf("\n\n");
    deleteNode(root,17);
    printf("\n After deleting 17 from tree :");
    printf("\n nodes in the tree are :");
    preorder(root);
    printf("\n\n");
    deleteNode(root,41);
    printf("\n After deleting 41 from tree :");
    printf("\n  nodes in the tree are :");
    printf("\n\n");
    preorder(root);
    printf("\n\n");
    getch();
    return 0;
}
```

Output:

```
Nodes  of the tree are: 18 15 17 40 30 50 41
 After inserting 45 and 19 nodes in tree :

18 15 17 40 30 19 50 41 45


 After deleting 15 from tree :
 nodes in the tree :18 17 40 30 19 50 41 45


 After deleting 17 from tree :
 nodes in the tree are :18 40 30 19 50 41 45


 After deleting 41 from tree :
  nodes in the tree are :

18 40 30 19 50 45

_
```

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.

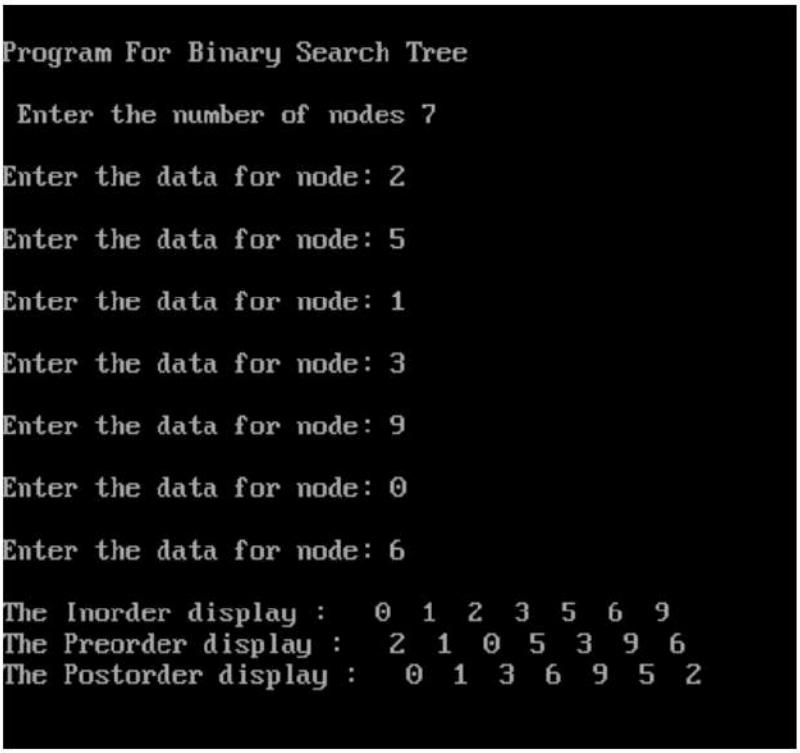
```
# include <stdio.h>
# include <conio.h>
# include <stdlib.h>
typedef struct BST
{
 int data;
 struct BST *lchild, *rchild;

} 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,i=1;
    node *new_node, *root;
    node *create_node();
    root = NULL;
    clrscr();
    printf("\nProgram For Binary Search Tree\n");
    printf("\n Enter the number of nodes");
    scanf("%d",&n);
    for(i=1;i<=n;i++)
       {
       new_node = create_node();
       printf("\nEnter the data for node:");
          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();
}
```

Output:



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();
}
```

Output:

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

```
# 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();
}
```

OUTPUT:

```
S1 = Flowers  S2 = are beautiful
1. Length 2.Concatenate 3.Extract Substring 4.REPLACE 5.Exit

Enter your choice : 1

Length of Flowers = 7_
```

```
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_
```

```
S1 = Flowers S2 = are beautiful
1. Length 2.Concatenate 3.Extract Substring 4.REPLACE 5.Exit

Enter your choice : 5
```