


1. Write a program for sorting 'n' elements using bubble sort technique.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int i,j,temp,n,a[20];
clrscr();
printf("enter size of array\n");
scanf("%d",&n);
printf("Enter elements\n");
for(i=0;i<n;i++)
{
scanf("%d", &a[i]);
}
printf("Elements before sorting are:\n");
for(i=0;i<n;i++)
{
printf("%d\t",a[i]);
}
printf("\nBubble sort\n");
for(i=0;i<n;i++)
{
for (j=0;j<n-i-1;j++)
{
if (a[j]>a[j+1])
{
temp=a[j];
a[j]=a[j+1];
a[j+1]=temp;
}
}
}
printf("Elements after sorting are:\n");
for(i=0;i<n;i++)
{
printf("%d\t",a[i]);
}
getch();
}
```

**Output:**



```
enter size of array
5
Enter elements
11
2
33
1
12
Elements before sorting are:
11    2    33    1    12
Bubble sort
Elements after sorting are:
1    2    11    12    33
```

2. Write a program to sort 'n' elements using selection sort.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int i,j,temp,n,a[20],min;
clrscr();
printf("enter size of array\n");
scanf("%d",&n);
printf("enter elements\n");
for(i=0;i<n;i++)
{
scanf("%d", &a[i]);
}
printf("Elements before sorting are:\n");
for(i=0;i<n;i++)
{
printf("%d\t",a[i]);
}
printf("\nSelection sort\n");
for(i=0;i<n;i++)
{
min=i;
for (j=i+1;j<n;j++)
{
if (a[j]<a[min])
min=j;
}
temp=a[i];
a[i]=a[min];
a[min]=temp;
}
printf("Elements after sorting are:\n");
for(i=0;i<n;i++)
{
printf("%d\t",a[i]);
}
getch();
}
```

**Output:**

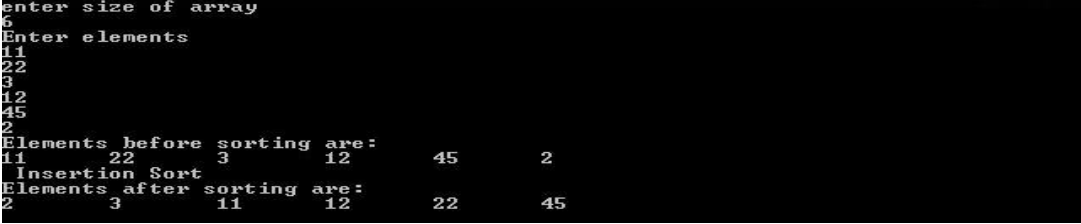


```
enter size of array
5
enter elements
111
2
12
33
45
Elements before sorting are:
111 2 12 33 45
Selection sort
Elements after sorting are:
2 12 33 45 111 _
```

**3. Write a program to sort elements using insertion sort.**

```
#include<stdio.h>
#include<conio.h>
void main()
{
int i,j,temp,n,a[20],index;
clrscr();
printf("enter size of array\n");
scanf("%d",&n);
printf("Enter elements\n");
for(i=0;i<n;i++)
{
scanf("%d", &a[i]);
}
printf("Elements before sorting are:\n");
for(i=0;i<n;i++)
{
printf("%d\t",a[i]);
}
printf("\n Insertion Sort\n");
for(i=1;i<n;i++)
{
index=a[i];
j=i;
while((j>0) && (a[j-1]>index))
{
a[j]=a[j-1];
j=j-1;
}
a[j]=index;
}
printf("Elements after sorting are:\n");
for(i=0;i<n;i++)
{
printf("%d\t",a[i]);
}
getch();
}
```

**Output:**



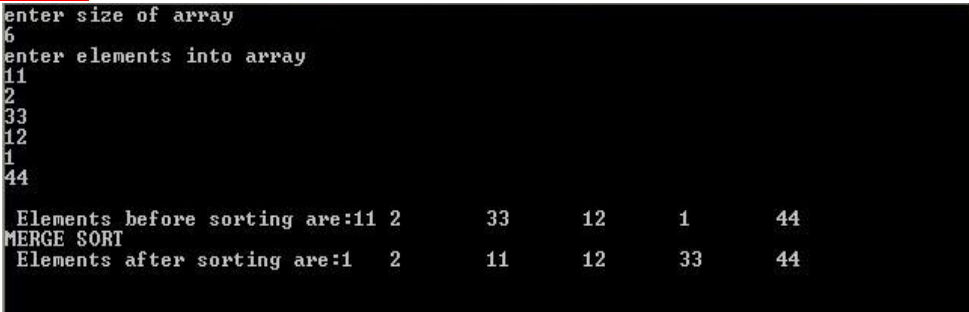
```
enter size of array
5
Enter elements
11
22
3
12
45
2
Elements before sorting are:
11    22    3    12    45    2
Insertion Sort
Elements after sorting are:
2    3    11    12    22    45
```

**4. Write a program to sort the elements using Merge sort.**

```
#include<stdio.h>
#include<conio.h>
void mergesort(int [],int ,int,int);
void mergepass(int [],int,int);
void main()
{
    int i,j,h,k,n,a[20];
    clrscr();
    printf("enter size of array\n");
    scanf("%d",&n);
    printf("enter elements into array\n");
    for(i=0;i<n;i++)
    {
        scanf("%d",&a[i]);
    }
    printf("\n Elements before sorting are:");
    for(i=0;i<n;i++)
    {
        printf("%d\t",a[i]);
    }
    printf("\nMERGE SORT");
    mergepass(a,0,n-1);
    printf("\n Elements after sorting are:");
    for (i=0;i<n;i++)
    {
        printf("%d\t",a[i]);
    }
    getch();
}
void mergepass(int a[],int low,int high)
{
    int mid;
    if(low!=high)
    {
        mid=(low+high)/2;
        mergepass(a,low,mid);
        mergepass(a,mid+1,high);
        mergesort(a,low,mid,high);
    }
}
void mergesort(int a[],int low,int mid,int high)
{
    int i,j,h,k,b[20];
    h=low;
    i=low;
    j=mid+1;
    while((h<=mid)&&(j<=high))
    {
```

```
        if(a[h]<=a[j])
        {
            b[i]=a[h];
            h=h+1;
        }
        else
        {
            b[i]=a[j];
            j=j+1;
        }
        i=i+1;
    }
    if(h<=mid)
    {
        for(h=h;h<=mid;h++)
        {
            b[i]=a[h];
            i=i+1;
        }
    }
    else
    {
        for(j=j;j<=high;j++)
        {
            b[i]=a[j];
            i=i+1;
        }
    }
    for(k=low;k<=high;k++)
        a[k]=b[k];
}
```

**Output:**



```
enter size of array
6
enter elements into array
11
2
33
12
1
44


Elements before sorting are:11 2      33      12      1      44
MERGE SORT
Elements after sorting are:1  2      11      12      33      44
```

**5. Write a program to sort the elements using Quick sort.**

```
#include<stdio.h>
#include<conio.h>
void quick(int [],int,int);
int partition(int,int);
int a[10];
void main()
{
    int n,i;
    clrscr();
    printf("Enter the size of array\n");
    scanf("%d",&n);
    printf("\n Enter elements into array\n");
    for(i=0;i<n;i++)
    {
        scanf("%d",&a[i]);
    }
    printf("\nelements before sorting\n");
    for(i=0;i<n;i++)
    {
        printf("%d\t",a[i]);
    }
    printf("\n Quick Sort\n");
    quick(a,0,n-1);
    printf("\nthe sorted elements are\n");
    for(i=0;i<n;i++)
    {
        printf("%d\t",a[i]);
    }
    getch();
}
void quick(int a[],int low ,int high)
{
    int pivot,i,j,temp;
    if(low<high)
    {
        i=low;
        j=high+1;
        pivot=a[low];
        do
        {
            do
                i++;
            while(a[i]<pivot);
            do
                j--;
            while(a[j]>pivot);
            if(i<j)
```

```
        temp=a[i];
        a[i]=a[j];
        a[j]=temp;
    }
}while(i<j);
temp=a[low];
a[low]=a[j];
a[j]=temp;
quick(a,low,j-1);
quick(a,j+1,high);
}
}
```

**Output:**



```
Enter the size of array
6
Enter elements into array
11
2
33
12
1
44
elements before sorting
11    2    33    12    1    44
Quick Sort
the sorted elements are
1    2    11    12    33    44    -
```

**6. Implement the following operations on single linked list.**

**a)Creation          b) Insertion          c)Deletion          4)Display**

```
#include<stdio.h>
#include<conio.h>
struct slist
{
    int el;
    struct slist * next;
};
typedef struct slist *list;
typedef struct slist *node;
typedef struct slist *position;
node create( );
position firstl(list);
position nextl(position );
position endl(list);
void read(list);
void insert(position,int);
void print(list);
void append(list,int);
void delet(list,position);
position prevl(list,position);
node head;

node create()
{
    node t,t1;
    t=(node)malloc(sizeof(node));
    t->el=0;
    t->next=NULL;
    return t;
}
void read(list l)
{
    int n,i,ele;
    position p;
    printf("\nEnter no of nodes u want to insert in list 1\n");
    scanf("%d",&n);
    for(i=0;i<n;i++)
    {
        printf("enter element\n");
        scanf("%d",&ele);
        append(l,ele);
    }
}

void append(list l,int n)
{
    insert(endl(l),n);
}
```



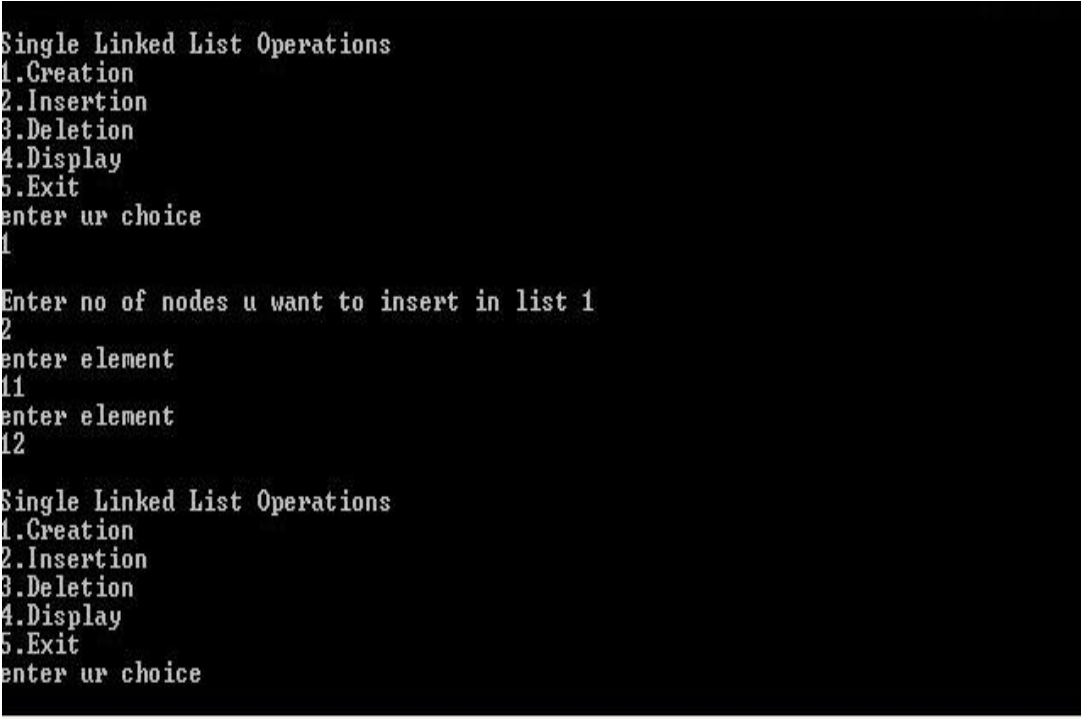
```
}
void insert(position p ,int n)
{
    node t;
    t=(node)malloc(sizeof(node));
    t->el=n;
    t->next=p->next;
    p->next=t;
}
void print(list l)
{
    position i;
    printf("\nElements in the list are:\n");
    for(i=firstl(l);i!=endl(l);i=nextl(i))
    {
        printf("%d\t",i->next->el);
    }
}
position firstl(list l)
{
    return l;
}
position nextl(position p)
{
    return p->next;
}
position endl(list l)
{
    list t;
    t=l;
    while(t->next!=NULL)
        t=t->next;
    return t;
}
position prevl(list l,position p)
{
    position t1,t;
    t=l;
    while(t!=p)
    {
        t1=t;
        t=t->next;
    }
    return t1;
}
void delet(list l,position p)
{
    position prev;
    prev=prevl(l,p);
    prev->next=p->next;
}
```

```
        free(p);
    }
    position find(list l)
    {
        position s;
        int n;
        s=l;
        printf("Enter element to find the position\n");
        scanf("%d",&n);
        while(s->el!=n && s->next!=NULL)
        {
            s=s->next;
        }
        if(s->el==n)
        {
            printf("\nElement found\n");
            return s;
        }
        else
        {
            printf("\nThere is no element\n");
            return 0;
        }
    }

    void main()
    {
        int n,m,ch;
        list l;
        position pos;
        clrscr();
        l=create();
        while(1)
        {
            printf("\nSingle Linked List Operations\n");
            printf("1.Creation\n2.Insertion\n3.Deletion\n4.Display\n5.Exit\n");
            printf("enter ur choice\n");
            scanf("%d",&ch);
            switch(ch)
            {
                case 1: read(l);
                        break;
                case 2: printf("Enter element to insert\n");
                        scanf("%d",&n);
                        pos=find(l);
                        insert(pos,n);
                        break;
                case 3: pos=find(l);
                        delet(l,pos);
            }
        }
    }
}
```

```
        break;
    case 4: print(l);
            break;
    case 5: exit(0);
            }
        }
    }
```

**Output:**



```
Single Linked List Operations
1.Creation
2.Insertion
3.Deletion
4.Display
5.Exit
enter ur choice
1
Enter no of nodes u want to insert in list 1
2
enter element
11
enter element
12

Single Linked List Operations
1.Creation
2.Insertion
3.Deletion
4.Display
5.Exit
enter ur choice
```

**7. Implement the following operations n double linked list.**

**a)Creation            b)Insertion    c) Deletion    d)Display**

```
#include<stdio.h>
#include<conio.h>
struct dlist
{
    int el;
    struct dlist * next;
    struct dlist *prev;
};
typedef struct dlist *list;
typedef struct dlist *node;
typedef struct dlist *position;
node create( );
position firstl(list);
position nextl(position );
position endl(list);
void read(list);
void insert(position,int);
void print(list);
void append(list,int);
void delet(list,position);
position prevl(list,position);
node head;
node create()
{
    node t,t1;
    t=(node)malloc(sizeof(node));
    t->el=0;
    t->next=NULL;
    t->prev=NULL;
    return t;
}
void read(list l)
{
    int n,i,ele;
    position p;
    printf("\nEnter no of nodes u want to insert in list 1\n");
    scanf("%d",&n);
    for(i=0;i<n;i++)
    {
        printf("enter element\n");
        scanf("%d",&ele);
        append(l,ele);
    }
}

void append(list l,int n)
{
```

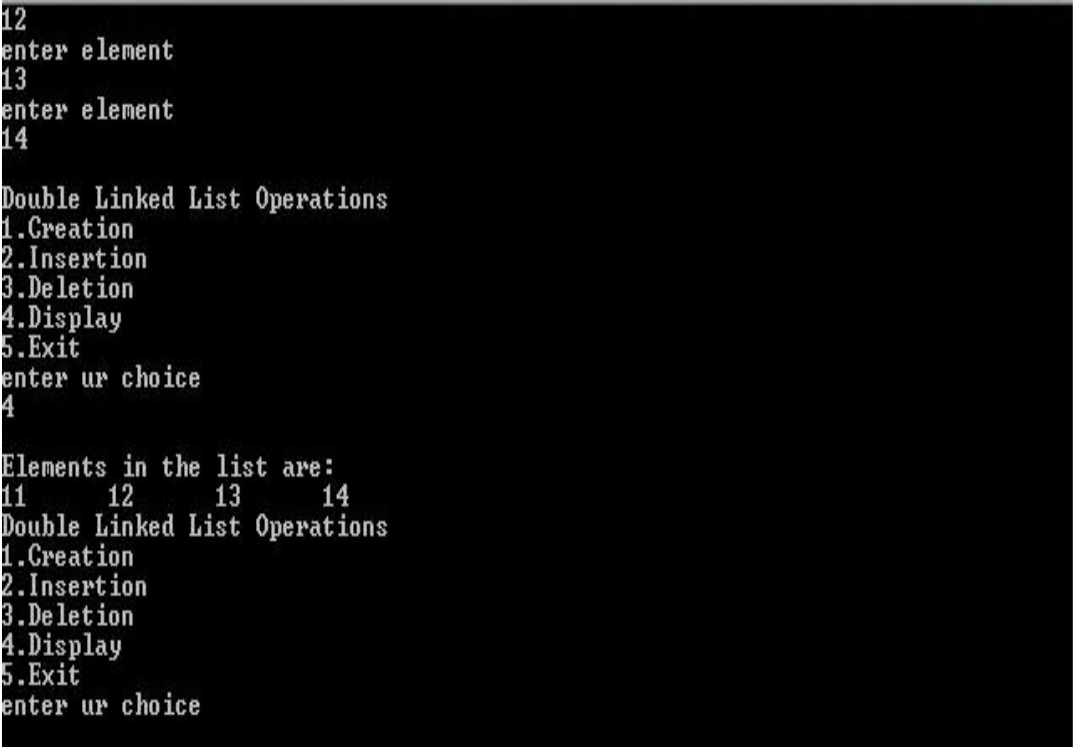
```
        insert(endl(l),n);
    }
void insert(position p ,int n)
{
    node t;
    t=(node)malloc(sizeof(node));
    t->el=n;
    t->next=p->next;
    t->prev=p;
    p->next=t;
    p->next->prev=t;
}
void print(list l)
{
    position i;
    printf("\nElements in the list are:\n");
    for(i=firstl(l);i!=endl(l);i=nextl(i))
    {
        printf("%d\t",i->next->el);
    }
}
position firstl(list l)
{
    return l;
}

position nextl(position p)
{
    return p->next;
}
position endl(list l)
{
    list t;
    t=l;
    while(t->next!=NULL)
        t=t->next;
    return t;
}
position prevl(list l,position p)
{
    position t1,t;
    t=l;
    while(t!=p)
    {
        t1=t;
        t=t->next;
    }
    return t1;
}
```

```
void delet(list l,position p)
{
    position prev;
    prev=prevl(l,p);
    prev->next=p->next;
    free(p);
}
position find(list l)
{
    position s;
    int n;
    s=l;
    printf("Enter element to find the position\n");
    scanf("%d",&n);
    while(s->el!=n && s->next!=NULL)
    {
        s=s->next;
    }
    if(s->el==n)
    {
        printf("\nElement found\n");
        return s;
    }
    else
    {
        printf("\nThere is no element\n");
        return 0;
    }
}
void main()
{
    int n,m,ch;
    list l;
    position pos;
    clrscr();
    l=create();
    while(1)
    {
        printf("\nDouble Linked List Operations\n");
        printf("1.Creation\n2.Insertion\n3.Deletion\n4.Display\n5.Exit\n");
        printf("enter ur choice\n");
        scanf("%d",&ch);
        switch(ch)
        {
            case 1: read(l);
                    break;
            case 2: printf("Enter element to insert\n");
                    scanf("%d",&n);
                    pos=find(l);
                    insert(pos,n);
                    break;
        }
    }
}
```

```
        case 3: pos=find(l);
                delet(l,pos);
                break;
        case 4: print(l);
                break;
        case 5:exit(0);
    }
}
```

**Output:**



```
12
enter element
13
enter element
14

Double Linked List Operations
1.Creation
2.Insertion
3.Deletion
4.Display
5.Exit
enter ur choice
4

Elements in the list are:
11    12    13    14
Double Linked List Operations
1.Creation
2.Insertion
3.Deletion
4.Display
5.Exit
enter ur choice
```

**8. Implement the following operations on circular linked list.**

**Creation      b)Insertion      c)Deletion      d) Display**

```
#include<stdio.h>
#include<conio.h>
struct clist
{
    int el;
    struct clist * next;
};
typedef struct clist * node;
typedef struct clist * list;
typedef struct clist * position;
void create(int n) ;
void insert(position p,int ele);
position find();
void display();
void delet(position);
position prev(position);
void read();
node last=NULL;
void read()
{
    int i,ele,sz;
    printf("Enter no of nodes u want to insert in the list\n");
    scanf("%d",&sz);
    for(i=0;i<sz;i++)
    {
        printf("enter element to insert\n");
        scanf("%d",&ele);
        create(ele);
    }
}
void create(int ele)
{
    node t;
    t=(node)malloc(sizeof(node));
    t->el=ele;
    t->next=NULL;
    if(last==NULL)
    {
        last=t;
        t->next=last;
    } else
    {
        t->next=last->next;
        last->next=t;
        last=t;
    }
}
```



```
void display()
{ node t;
if(last==NULL)
{
printf("List is empty\n"); }
t=last->next;
while(t!=last)
{
printf("%d\t",t->el);
t=t->next;
}
printf("%d\t",last->el);
}
void insert(position p,int ele)
{
node t;
t=(node)malloc(sizeof(node));
t->el=ele; t->next=p->next;
p->next=t;
if(p==last) last=t;
}
position find ()
{
position s;
int n; s=last;
printf("Enter element to find the position\n");
scanf("%d",&n);
while(s->el!=n && s->next!=last)
{ s=s->next;
}
if(s->el==n)
{
printf("\nElement found\n");
return s;
}
else
{
printf("\nThere is no element\n");
return 0;
}
}

position prevl(position p)
{ position t1,t;
t=last;
while(t!=p)
{ t1=t;
t=t->next;
} return t1;
}
void delet(position p)
{
```

```
position prev;
prev=prevl(p);
prev->next=p->next;
free(p);
}
void main()
{
int ch,n;
position pos;
list l; //l=create();
clrscr();
while(1)
{
printf("\n Circular Linked List Operation\n");
printf("1.Creation\n2.Display\n3.Insertion\n4.Deletion\n5.Exit\n");
printf("Enter ur choice\n");
scanf("%d",&ch)

switch(ch)
{
case 1:read();
break;
case 2:display();
break;
case 3:pos=find();
printf("Enter element to insert\n");
scanf("%d",&n); insert(pos,n);
break;
case 4:pos=find();
if(pos!=0) delet(pos);
else
printf("there is no element\n");
break;
case 5:exit(0);
}

}
}
```

Output

Output:

```
1
Enter no of nodes u want to insert in the list
2
enter element to insert
11
enter element to insert
12
```

```
    Circular Linked List Operation
1.Creation
2.Display
3.Insertion
4.Deletion
5.Exit
Enter ur choice
```

```
2
11      12
    Circular Linked List Operation
1.Creation
2.Display
3.Insertion
4.Deletion
5.Exit
Enter ur choice
```

**9. Program for splitting given linked list.**

```
#include<stdio.h>
#include<conio.h>
struct slist
{
    int el;
    struct slist * next;
};
typedef struct slist *list;
typedef struct slist *node;
typedef struct slist *position;
node create( );
position firstl(list);
position nextl(position );
position endl(list);
void read(list);
void insert(position,int);
void print(list);
void append(list,int);
void split(list);
node head;
node create()
{
    node t,t1;
    t=(node)malloc(sizeof(node));
    t->el=0;
    t->next=NULL;
    return t;
}
void read(list l)
{
    int n,i,ele;
    position p;
    printf("\nEnter no of nodes u want to insert in list 1\n");
    scanf("%d",&n);
    for(i=0;i<n;i++)
    {
        printf("enter element\n");
        scanf("%d",&ele);
        append(l,ele);
    }
}
void append(list l,int n)
{
    insert(endl(l),n);
}
void insert(position p ,int n)
{
    node t;
```

```
        t=(node)malloc(sizeof(node));
        t->el=n;
        t->next=p->next;
        p->next=t;
    }

void print(list l)
{
    position i;
    printf("\nElements in the list are:\n");
    for(i=firstl(l);i!=endl(l);i=nextl(i))
    {
        printf("%d\t",i->next->el);
    }
}

position firstl(list l)
{
    return l;
}

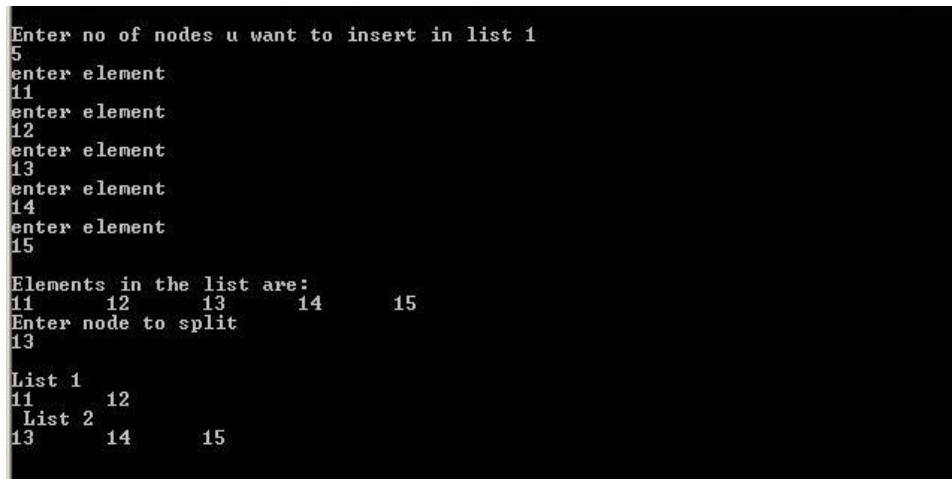
position nextl(position p)
{
    return p->next;
}

position endl(list l)
{
    list t;
    t=l;
    while(t->next!=NULL)
        t=t->next;
    return t;
}

void split(list l)
{
    int ele;
    node t,t1;
    printf("\nEnter node to split\n");
    scanf("%d",&ele);
    t=l;
    while(t->el!=ele)
    {
        t1=t;
        t=t->next;
    }
    t1->next=NULL;
    t1=t;
    t=l;
    printf("\nList 1 \n");
    while(t->next!=NULL)
    {
        t=t->next;
```

```
        printf("%d\t", t->el);
    }
    printf("\n List 2\n");
    t=t1;
    while(t!=NULL)
    {
        printf("%d\t",t->el);
        t=t->next;
    }
}
void main()
{
    list l;
    clrscr();
    l=create();
    read(l);
    print(l);
    split(l);
    getch();
}
```

**Output:**



```
Enter no of nodes u want to insert in list 1
5
enter element
11
enter element
12
enter element
13
enter element
14
enter element
15

Elements in the list are:
11    12    13    14    15
Enter node to split
13

List 1
11    12
List 2
13    14    15
```

**10. Program for traversing the given linked list in reverse order.**

```
#include<stdio.h>
#include<conio.h>
struct slist
{
    int el;
    struct slist * next;
};
typedef struct slist *list;
typedef struct slist *node;
typedef struct slist *position;
node create( );
position firstl(list);
position nextl(position );
position endl(list);
void read(list);
void insert(position,int);
void print(list);
void append(list,int);
position prevl(list,position);
void reverse(list);
node create()
{
    node t,t1;
    t=(node)malloc(sizeof(node));
    t->el=0;
    t->next=NULL;
    return t;
}
void read(list l)
{
    int n,i,ele;
    position p;
    printf("\nEnter no of nodes u want to insert in list 1\n");
    scanf("%d",&n);
    for(i=0;i<n;i++)
    {
        printf("enter element\n");
        scanf("%d",&ele);
        append(l,ele);
    }
}
void append(list l,int n)
{
    insert(endl(l),n);
}
void insert(position p ,int n)
{
```

```
        node t;
        t=(node)malloc(sizeof(node));
        t->el=n;
        t->next=p->next;
        p->next=t;
    }

void print(list l)
{
    position i;
    printf("\nElements in the list are:\n");
    for(i=firstl(l);i!=endl(l);i=nextl(i))
    {
        printf("%d\t",i->next->el);
    }
}

position firstl(list l)
{
    return l;
}

position nextl(position p)
{
    return p->next;
}

position endl(list l)
{
    list t;
    t=l;
    while(t->next!=NULL)
        t=t->next;
    return t;
}

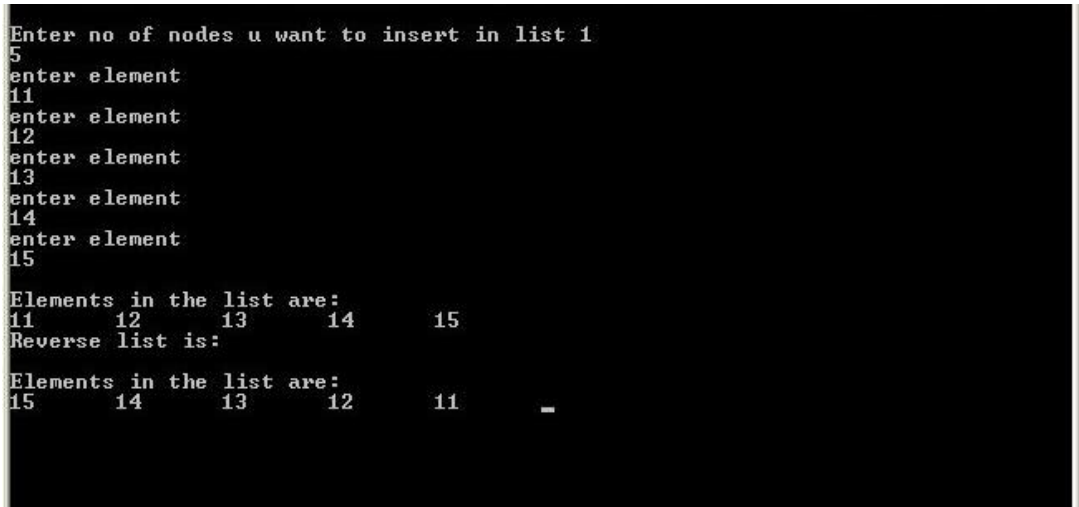
position prevl(list l,position p)
{
    position t1,t;
    t=l;
    while(t!=p)
    {
        t1=t;
        t=t->next;
    }
    return t1;
}

void reverse(list l)
{
    position i,j;
    i=firstl(l);
    j=endl(l);
    i=nextl(i);
    for(;i!=j&& i!=nextl(j);i=nextl(i),j=prevl(l,j))
```



```
        {
            int t;
            t=i->el;
            i->el=j->el;
            j->el=t;
        }
    }
}
void main()
{
    int n,m,ch;
    list l,l1,l2,l3;
    position pos;
    clrscr();
    l=create();
    read(l);
    print(l);
    printf("\nReverse list is:\n");
    reverse(l);
    print(l);
    getch();
}
```

**Output:**



```
Enter no of nodes u want to insert in list 1
5
enter element
11
enter element
12
enter element
13
enter element
14
enter element
15

Elements in the list are:
11    12    13    14    15
Reverse list is:

Elements in the list are:
15    14    13    12    11    -
```

**11. Merge two given linked lists.**

```
#include<stdio.h>
#include<conio.h>
struct slist
{
    int el;
    struct slist * next;
};
typedef struct slist *list;
typedef struct slist *node;
typedef struct slist *position;
node create( );
position firstl(list);
position nextl(position );
position endl(list);
void read(list);
void insert(position,int);
void print(list);
void merge(list,list,list);
void append(list,int);
node head;
node create()
{
    node t,t1;
    t=(node)malloc(sizeof(node));
    t->el=0;
    t->next=NULL;
    return t;
}

void read(list l)
{
    int n,i,ele;
    position p;
    printf("\nEnter no of nodes u want to insert in list\n");
    scanf("%d",&n);
    for(i=0;i<n;i++)
    {
        printf("enter element\n");
        scanf("%d",&ele);
        append(l,ele);
    }
}

void append(list l,int n)
{
    insert(endl(l),n);
}
```

```
}
void insert(position p ,int n)
{
    node t;
    t=(node)malloc(sizeof(node));
    t->el=n;
    t->next=p->next;
    p->next=t;
}
void print(list l)
{
    position i;
    printf("\nElements in the list are:\n");
    for(i=firstl(l);i!=endl(l);i=nextl(i))
    {
        printf("%d\t",i->next->el);
    }
}
position firstl(list l)
{
    return l;
}
position nextl(position p)
{
    return p->next;
}
position endl(list l)
{
    list t;
    t=l;
    while(t->next!=NULL)
        t=t->next;
    return t;
}
position prevl(list l,position p)
{
    position t1,t;
    t=l;
    while(t!=p)
    {
        t1=t;
        t=t->next;
    }
    return t1;
}
void merge(list l1,list l2,list l3)
{
    position i,j;
    i=firstl(l1);
    j=firstl(l2);
    while(i!=endl(l1) && j!=endl(l2))
```

```
    {
        if( i->next->el<j->next->el)
        {
            append(l3,i->next->el);
            i=nextl(i);
        }
        else if(i->next->el>j->next->el)
        {
            append(l3,j->next->el);
            j=nextl(j);
        }
        else
        {
            append(l3,i->next->el);
            i=nextl(i);
            j=nextl(j);
        }
        while(i!=endl(l1))
        {
            append(l3,i->next->el);
            i=nextl(i);
        }
        while(j!=endl(l2))
        {
            append(l3,j->next->el);
            j=nextl(j);
        }
    }
}
void main()
{
    list l1,l2,l3;
    clrscr();
    l1=create();
    l2=create();
    l3=create();
    printf("\nList1\n");
    read(l1);
    print(l1);
    printf("\nList2\n");
    read(l2);
    print(l2);
    printf("\nMerged list is:\n");
    merge(l1,l2,l3);
    print(l3);
    getch();
}
```

**Output:**

```
Enter no of nodes u want to insert in list
2
enter element
11
enter element
12
Elements in the list are:
11    12
List2
Enter no of nodes u want to insert in list
2
enter element
45
enter element
56
Elements in the list are:
45    56
Merged list is:
Elements in the list are:
11    12    45    56
```

**12. Create a linked list to store the names of colors.**

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
struct color
{
    char data[120];
    struct color * next;
};
typedef struct color *list;
void read(int );
void display();
list head=NULL,last=NULL;
void read(int n)
{
    list t,t1;
    int i;
    char a[120];
    for(i=0;i<n;i++)
    {
        printf("enter name of color\n");
        scanf("%s",a);
        t=(list)malloc(sizeof(list));
        strcpy(t->data,a);
        t->next=NULL;
        if(head==NULL)
            head=last=t;
        else
        {
            last->next=t;
            last=t;
        }
    }
}
void display()
{
    list t;
    t=head;
    printf("Colors in the list are\n");
    while(t!=NULL)
    {
        printf("%s\n",t->data);
        t=t->next;
    }
}
void main()
{
    list l;
```

```
int n;  
clrscr();  
printf("Enter no of nodes to insert\n");  
scanf("%d",&n);  
read(n);  
display();  
getch();  
}
```

**Output:**

```
Enter no of nodes to insert  
3  
enter name of color  
red  
enter name of color  
cyan  
enter name of color  
pink  
Colors in the list are  
red  
cyan  
pink  
-
```

**13. Implement stack operations using arrays.**

```
#include<stdio.h>
#include<conio.h>
#define sz 5
typedef int element;
struct stack
{
    int top;
    element *a;
};
typedef struct stack *st;
st create();
void push(st,int);
void display(st);
int pop(st);
int isempty(st);
int tops(st s);
int isfull(st );
st create()
{
    st s;
    s=(st)malloc(sizeof(st));
    s->top=-1;
    s->a=(int *)malloc(sizeof(element)*sz);
    return s;
}
void push(st s,int el)
{
    if(isfull(s))
    {
        printf("stack is full\n");
    }
    else
    {
        s->top++;
        s->a[s->top]=el;
    }
}
int isfull(st s)
{
    if(s->top>=sz-1)
        return 1;
    else
        return 0;
}
void display(st s)
{
    int i;
```



```

        if(isempty(s))
        {
            printf("Stack is empty\n");
        }
        else
        {
            printf("\nElements are:\n");
            for(i=s->top;i>=0;i--)
            {
                printf("%d\t",s->a[i]);
            }
        }
    }
int tops(st s)
{
    return s->a[s->top];
}
int pop(st s)
{
    int el;
    if(isempty(s))
    {
        printf("Stack is empty\n");
    }
    else
    {
        el=s->a[s->top];
        s->top--;
    }
    return el;
}
int isempty(st s)
{
    if(s->top== -1)
        return 1;
    else
        return 0;
}
void main()
{
    st s;
    int el,ch;
    s=create();
    clrscr();
    while(1)
    {
        printf("\nStack Operations\n");
        printf("1.Push\n2.Pop\n3.Top\n4.Exit\n");
        printf("Enter ur choice\n");
        scanf("%d",&ch);
    }
}

```

```
switch(ch)
{
    case 1:printf("enter element to insert\n");
           scanf("%d",&el);
           push(s,el);
           display(s);
           break;
    case 2:if(isempty(s))
           {
               printf("Stack is empty,there are no elements\n");
           }
           else
           {
               el=pop(s);
               printf("Deleted element is %d:\t",el);
           }
           display(s);
           break;
    case 3:if(isempty(s))
           {
               printf("there is no top element\n");
           }
           else
           {
               el=tops(s);
               printf("Top element is %d\t", el);
           }
           break;
    case 4:exit(0);
}
}
```

**Output:**

```
1
enter element to insert
11
Elements are:
11
Stack Operations
1.Push
2.Pop
3.Top
4.Exit
Enter ur choice
1
enter element to insert
12
Elements are:
12 11
Stack Operations
1.Push
2.Pop
3.Top
4.Exit
Enter ur choice
```

**14. Implement Stack operations using linked list.**

```
#include<stdio.h>
#include<conio.h>
struct stack
{
    int el;
    struct stack *next;
};
typedef struct stack * node;
node head=NULL;
node top=NULL;
void push(int);
int pop();
void display();
node create();
int tops();
node create()
{
    node temp;
    temp=(node)malloc(sizeof(node));
    temp->el=0;
    temp->next=NULL;
    return temp;
}
void push(int n)
{
    node temp;
    temp=create();
    temp->el=n;
    temp->next=top;
    top=temp;
    printf("Data pushed into stack\n");
}
int pop()
{
    node temp;
    int n;
    if(top==NULL)
    {
        printf("stack is empty\n");
    }
    else
    {
        temp=top;
        n=temp->el;
        top=temp->next;
        free(temp);
    }
}
```

```

        return n;
    }
    void display()
    {
        node temp;
        if(top==NULL)
        {
            printf("Stack is empty\n");
        }
        else
        {
            temp=top;
            printf("Elements in stack are:\n");
            while(temp!=NULL)
            {
                printf("%d\t",temp->el);
                temp=temp->next;
            }
        }
    }
    int tops()
    {
        return top->el;
    }

    void main()
    {
        int ch,n;
        node t;
        clrscr();
        while(1)
        {
            printf("\nStack operations using linked list\n");
            printf("1.Push\n2.Pop\n3.Top\n4.Exit\n");
            printf("Enter ur choice\n");
            scanf("%d",&ch);
            switch(ch)
            {
                case 1: printf("enter element to insert\n");
                        scanf("%d",&n);
                        push(n);
                        display();
                        break;
                case 2: n=pop();
                        printf("poped element is %d\t",n);
                        display();
                        break;
                case 3:n=tops();
                        printf("Top element is %d" , n);
                        break;
            }
        }
    }
}

```

```
                case 4:exit(0);  
            }  
        }  
    }
```

**Output:**

```
1  
enter element to insert  
11  
Data pushed into stack  
Elements in stack are:  
11  
Stack operations using linked list  
1.Push  
2.Pop  
3.Top  
4.Exit  
Enter ur choice  
1  
enter element to insert  
12  
Data pushed into stack  
Elements in stack are:  
12    11  
Stack operations using linked list  
1.Push  
2.Pop  
3.Top  
4.Exit  
Enter ur choice
```

**15. Implement Queue operations using arrays.**

```

#include<stdio.h>
#include<conio.h>
#define sz 5
typedef int element;
struct queue
{
    int rear;
    int front;
    element *a;
};
typedef struct queue *qu;
void addQ(qu,int );
int deleteQ(qu);
int isfull(qu);
int isempty(qu);
void display(qu);
qu create();
void main()
{
    int n,ch,el;
    qu q;
    q=create();
    clrscr();
    while(1)
    {
        printf("\nQueue Operations\n");
        printf("1.AddQ\n2.DeleteQ\n3.Display\n4.Exit\n");
        printf("enter ur choice\n");
        scanf("%d",&ch);
        switch(ch)
        {
            case 1:printf("enter element\n");
                scanf("%d",&n);
                addQ(q,n);
                break;
            case 2: if(isempty(q))
                {
                    printf("queue is empty\n");
                }
                else
                {
                    el=deleteQ(q);
                    printf("Deleted element is %d:",el);
                }
                break;
            case 3: display(q);
                break;
            case 4:exit(0);
        }
    }
}

```

```

    }
}
qu create()
{
    qu q;
    q=(qu)malloc(sizeof(qu));
    q->rear=-1;
    q->front=-1;
    q->a=(int*)malloc(sizeof(element)*sz);
    return q;
}
void addQ(qu q,int el)
{
    if(isfull(q))
    {
        printf("Queue is full");
    }
    else
    {
        q->a[q->rear]=el;
        q->rear++;
    }
}

int deleteQ(qu q)
{
    int el;
    if(isempty(q))
    {
        printf("Queue is empty\n");
    }
    else
    {
        el=q->a[q->front];
        q->front++;
    }
    return el;
}
int isfull(qu q)
{
    if(q->rear==sz-1)
        return 1;
    else
        return 0;
}
int isempty(qu q)
{
    if(q->rear==q->front)
        return 1;
    else

```

```
        return 0;
    }
    void display(qu q)
    {
        int i;
        for(i=q->front;i<q->rear;i++)
        {
            printf("%d\t",q->a[i]);
        }
    }
}
```

**Output:**



```
Queue Operations
1.AddQ
2.DeleteQ
3.Display
4.Exit
enter ur choice
1
enter element
12
Queue Operations
1.AddQ
2.DeleteQ
3.Display
4.Exit
enter ur choice
3
11      12
Queue Operations
1.AddQ
2.DeleteQ
3.Display
4.Exit
enter ur choice
```



**16. Implement Queue operations using linked list.**

```
#include<stdio.h>
#include<conio.h>
struct queue
{
    int el;
    struct queue *next;
};
typedef struct queue *node;
node create();
void addq(int);
void deleteq();
void display();
node rear=NULL;
node front=NULL;
node create()
{
    node temp;
    temp=(node)malloc(sizeof(node));
    temp->el=0;
    temp->next=NULL;
    return temp;
}
void addq(int n)
{
    node t;
    t=create();
    t->el=n;
    t->next=NULL;
    if(front!=NULL)
        rear->next=t;
    else
        front=t;
    rear=t;
}
void deleteq()
{
    node t;
    int n;
    if(front==NULL)
    {
        printf("Queue is empty\n");
    }
    else
    {
        t=front;
        n=t->el;
        front=t->next;
```

```

        free(t);
        printf("Deleted element is %d \t",n);

    }

}
void display()
{
    node t;
    if(front==NULL)
    {
        printf("Queue is empty\n");
        return ;
    }
    else
    {
        t=front;
        printf("\nElement are:\n");
        while(t!=NULL)
        {
            printf("%d\t",t->el);
            t=t->next;
        }
    }
}
void main()
{
    int n,ch;
    clrscr();
    while(1)
    {
        printf("\nQueue Operations using Linked List\n");
        printf("1.AddQ\n2.DeleteQ\n3.Exit\n");
        printf("Enter ur choice\n");
        scanf("%d",&ch);
        switch(ch)
        {
            case 1:printf("Enter element to insert\n");
                    scanf("%d",&n);
                    addq(n);
                    display();
                    break;
            case 2:deleteq();
                    display();
                    break;
            case 3:exit(0);
        }
    }
}

```

**Output:**

```
3.Exit
Enter ur choice
1
Enter element to insert
11
Element are:
11
Queue Operations using Linked List
1.AddQ
2.DeleteQ
3.Exit
Enter ur choice
1
Enter element to insert
12
Element are:
11    12
Queue Operations using Linked List
1.AddQ
2.DeleteQ
3.Exit
Enter ur choice
```

**17. Implement Operations on circular queue.**

```

#include<stdio.h>
#include<conio.h>
#define sz 5
typedef int element;
struct cqueue
{
    int rear;
    int front;
    int *a;
};
typedef struct cqueue *cq;
cq create();
void addQ(cq,int );
int deleteQ(cq);
void display(cq);
void main()
{
    int n,ch,el;
    cq c;
    c=create();
    clrscr();
    while(1)
    {
        printf("\nQueue Operations\n");
        printf("1.AddQ\n2.DeleteQ\n3.Display\n4.Exit\n");
        printf("enter ur choice\n");
        scanf("%d",&ch);
        switch(ch)
        {
            case 1:printf("enter element\n");
                scanf("%d",&n);
                addQ(c,n);
                break;
            case 2: if(c->front==-1)
                    {
                        printf("Queue is empty\n");
                    }
                    else
                    {
                        el=deleteQ(c);
                        printf("Deleted element is %d",el);
                    }
                break;
            case 3: display(c);
                break;
            case 4:exit(0);
        }
    }
}

```

```

}
cq create()
{
    cq c;
    c=(cq)malloc(sizeof(cq));
    c->rear=-1;
    c->front=-1;
    c->a=(int *)malloc(sizeof(element)*sz);
    return c;
}
void addQ(cq c,int el)
{
    int r;
    if(c->front==0 && c->rear==sz-1 || c->rear+1==c->front)
    {
        printf("queue is full\n");
    }
    else
    {
        if(c->rear==sz-1)
            c->rear=0;
        else
            c->rear=(c->rear+1)%sz;
        c->a[c->rear]=el;
        if(c->front==-1)
            c->front=0;
    }
}
int deleteQ(cq c)
{
    int el;
    if(c->front==-1)
    {
        printf("Queue is empty\n");
    }
    else
    {
        //front=(front+1)%sz;
        el=c->a[c->front];
        if(c->front==c->rear)
            c->front=c->rear=-1;
        else
        {
            if(c->front==sz-1)
                c->front=0;
            else
                c->front=(c->front+1)%sz;
        }
    }
    return el;
}

```

```

void display(cq c)
{
    int i;
    if(c->front==-1&& c->rear==-1 )
    {
        printf("Queue is empty\n");
    }
    else
    {
        printf("Elements are\n");
        if(c->rear<c->front)
        {
            for(i=c->front;i<=sz-1;i++)
                printf("%d\t",c->a[i]);
            for(i=0;i<=c->rear;i++)
                printf("%d\t",c->a[i]);
        }
        else
        {
            for(i=c->front;i<=c->rear;i++)
                printf("%d\t",c->a[i]);
        }
    }
}

```

**Output:**

```

Queue Operations
1.AddQ
2.DeleteQ
3.Display
4.Exit
enter ur choice
1
enter element
12

Queue Operations
1.AddQ
2.DeleteQ
3.Display
4.Exit
enter ur choice
3
11    12
Queue Operations
1.AddQ
2.DeleteQ
3.Display
4.Exit
enter ur choice

```

**18. Implement Operations on double ended queue.**

```

#include<stdio.h>
#include<conio.h>
#define sz 5
typedef int element;
struct dqueue
{
    int rear;
    int front;
    element *a;
};
typedef struct dqueue *dq;
dq create();
void pushdq(dq,int);
void inject(dq,int);
int eject(dq);
int popdq(dq);
void display(dq);
int count=0;
void main()
{
    int ch,n,e;
    dq q;
    clrscr();
    q=create();
    while(1)
    {
        printf("Double Ended Queue\n");
        printf("1.Pushdq\n2.Inject\n3.Popdq\n4.Eject\n5.Display\n6.Exit\n");
        printf("enter ur choice\n");
        scanf("%d",&ch);
        switch(ch)
        {
            case 1:printf("enter element\n");
                scanf("%d",&e);
                pushdq(q,e);
                break;
            case 2:printf("enter element\n");
                scanf("%d",&e);
                inject(q,e);
                break;
            case 3:n=popdq(q);
                printf("deleted element is %d",n);
                break;
            case 4:n=eject(q);
                printf("Deleted element is %d",n);
                break;
            case 5: display(q);
                break;
        }
    }
}

```





```

    }
    else
    {
        q->front=s;
        q->a[q->front]=el;
    }
    */
}
void display(dq q)
{
    int i;
    if(count==0)
        printf("queue is empty\n");
    else
    {
        if(q->rear>=q->front)
        {
            for(i=q->front;i<=q->rear;i++)
            {
                printf("%d\t",q->a[i]);
            }
        }
        else
        {
            for(i=q->front;i<=sz-1;i++)
                printf("%d\t",q->a[i]);
            for(i=0;i<=q->rear;i++)
                printf("%d\t",q->a[i]);
        }
    }
}
void
inject(dq q,int el)
{
    if(q->rear==-1 && q->front==-1)
        q->rear=q->front=0;
    else
    {
        if(q->rear==sz-1)
            q->rear=0;
        else
            q->rear++;
    }
    q->a[q->rear]=el;
    count++;
}
int popdq(dq q)
{
    int el;
    if(count==0)

```

```

        printf("Queue is empty\n");
    else
    {
        el=q->a[q->front];
        if(q->front==sz-1)
            q->front=0;
        else
            q->front++;
        count--;
    }
    return el;
}
int eject(dq q)
{
    int el;
    el=q->a[q->rear];
    if(q->rear==0)
        q->rear=sz-1;
    else
        q->rear--;
    count--;
}
return el;
}

```

**Output:**

```

4.Eject
5.Display
6.Exit
enter ur choice
2
enter element
23
Double Ended Queue
1.Pushdq
2.Inject
3.Popdq
4.Eject
5.Display
6.Exit
enter ur choice
5
12      11      23      Double Ended Queue
1.Pushdq
2.Inject
3.Popdq
4.Eject
5.Display
6.Exit
enter ur choice

```

**19. Convert infix expression into postfix expression using stack.**

```

#include<stdio.h>
#include<conio.h>
#define sz 5
typedef char element;
struct postfix
{
    int top;
    element *a;
};
typedef struct postfix *post;
post create();
int isfull(post);
int isempty(post);
void push(post,element);
element pop(post);
element tops(post);
int pcd(char c);
void infix(post,element[]);
post create()
{
    post p;
    p=(post)malloc(sizeof(post));
    p->top=-1;
    p->a=(char*)malloc(sizeof(element)*sz);
    return p;
}
void push(post p,element el)
{
    if(isfull(p))
        printf("stack is full\n");
    else
    {
        p->top++;
        p->a[p->top]=el;
    }
}
element pop(post p)
{
    element el;
    if(isempty(p))
        printf("Stack is empty\n");
    else
    {
        el=p->a[p->top];
        p->top--;
    }
    return el;
}

```

```

int isempty(post p)
{
    if(p->top== -1)
        return 1;
    else
        return 0;
}
int isfull(post p)
{
    if(p->top==sz-1)
        return 1;
    else
        return 0;
}
element tops(post p)
{
    return p->a[p->top];
}
int pcd(char c)
{
    if(c=='$')
        return 1;
    else if(c=='(')
        return 2;
    else if(c=='+' || c=='-')
        return 3;
    else if(c=='*' || c=='/')
        return 4;
    else
        return 5;
}
void infix(post p,element c[])
{
    element x='$';
    int i;
    push(p,x);
    for(i=0;c[i]!='\0';i++)
    {
        if(c[i]=='(')
            push(p,c[i]);
        else if(c[i]==')')
        {
            while(tops(p)!='(')
                printf("%c",pop(p));
            pop(p);
        }
        else if(c[i]>=97 && c[i]<=122)
        {
            printf("%c",c[i]);
        }
        else
    }
}

```

```

        {
            if(pcd(c[i])>pcd(tops(p)))
            {
                push(p,c[i]);
            }
            else
            {
                while(pcd(c[i])<=pcd(tops(p)))
                {
                    printf("%c",pop(p));
                }
                push(p,c[i]);
            }
        }
        while(tops(p)!='$')
        printf("%c",pop(p));
    }
void main()
{
    int r,i;
    post p;
    element ch[30];
    clrscr();
    p=create();
    printf("enter ur expression\n");
    scanf("%s",ch);
    infix(p,ch);
    getch();
}

```

**Output:**

```

enter ur expression
a+b*c-d/e
abc**de/-

```

**20. Write a program to evaluate postfix expression.**

```

#include<stdio.h>
#include<conio.h>
#define sz 5
struct postfix
{
    int top;
    int *a;
};
typedef struct postfix *post;
post create();
int isfull(post);
int isempty(post);
void push(post,int);
int pop(post);
int tops(post);
int pcd(char c);
void eval(post,char);
post create()
{
    post p;
    p=(post)malloc(sizeof(post));
    p->top=-1;
    p->a=(int*)malloc(sizeof(int)*sz);
    return p;
}
void push(post p,int el)
{
    if(isfull(p))
        printf("stack is full\n");
    else
    {
        p->top++;
        p->a[p->top]=el;
    }
}
int pop(post p)
{
    int el;
    if(isempty(p))
        printf("Stack is empty\n");
    else
    {
        el=p->a[p->top];
        p->top--;
    }
    return el;
}

```

```

int isempty(post p)
{
    if(p->top== -1)
        return 1;
    else
        return 0;
}
int isfull(post p)
{
    if(p->top==sz-1)
        return 1;
    else
        return 0;
}
int tops(post p)
{
    return p->a[p->top];
}
int pcd(char c)
{
    if(c=='+')
        return 1;
    else if (c=='-')
        return 2;
    else if(c=='*')
        return 3;
    else if (c=='/')
        return 4;
    else if (c=='%')
        return 5;
    else if(c=='^')
        return 6;

    else
        return 7;
}
void eval(post p,char c)
{
    if(c>=48 && c<=57)
    {
        push(p,c-48);
    }
    else
    {
        int a,b,ch,n;
        a=pop(p);
        b=pop(p);
        ch= pcd(c);
        switch(ch)
        {
            case 1:n=a+b;

```

```
        push(p,n);
        break;
    case 2:n=a-b;
        push(p,n);
        break;
    case 3:n=a*b;
        push(p,n);
        break;
    case 4:n=a/b;
        push(p,n);
        break;
    case 5:n=a%b;
        push(p,n);
        break;
    case 6:n=a^b;
        push(p,n);
        break;
    case 7:exit(0);
}
}
}
void main()
{
    int r,i;
    post p;
    char ch[30];
    clrscr();
    p=create();
    printf("enter ur expression\n");
    scanf("%s",ch);
    for(i=0;ch[i]!='\0';i++)
        eval(p,ch[i]);
    r=pop(p);
    printf("Result is %d\t",r);
    getch();
}
```

**Output:**

```
enter ur expression
234**
Result is 14
```



**21. Implement operations on two way stack.**

```

#include<stdio.h>
#include<conio.h>
#define sz 5
struct stack
{
    int top1;
    int top2;
    int *a;
};
typedef struct stack * st;
st create();
void push(st);
int pop(st);
void print(st);
void main()
{
    int ch;
    st s;
    clrscr();
    s=create();
    while(1)
    {
        printf("\nTwo Way Stack Operations\n");
        printf("1.Push\n2.Pop\n3.Print\n4.Exit\n");
        printf("Enter ur choice\n");
        scanf("%d",&ch);
        switch(ch)
        {
            case 1: push(s);
                    break;
            case 2: pop(s);
                    break;
            case 3: print(s);
                    break;
            case 4: exit(0);
        }
    }
}
st create()
{
    st s;
    s=(st)malloc(sizeof(st));
    s->top1=-1;
    s->top2=sz;
    s->a=(int*)malloc(sizeof(int)*sz);
    return s;
}

```

```

void push(st s)
{
    int el,op;
    printf("\nInsert element into Stack1/Stack2\n");
    scanf("%d",&op);
    switch(op)
    {
        case 1:if(s->top1>=s->top2-1)
            {
                printf("Stack is full\n");
                break;
            }
        else
            {
                s->top1++;
                printf("Enter element to insert into stack1\n");
                scanf("%d",&el);
                s->a[s->top1]=el;
            }
        break;
        case 2:if(s->top2<s->top1+1)
            {
                printf("Stack is full\n");
                break;
            }
        else
            {
                s->top2--;
                printf("Enter element to insert into stack2\n");
                scanf("%d",&el);
                s->a[s->top2]=el;
            }
        break;
    }
}

int pop(st s)
{
    int el,op;
    printf("Delete element from Stack1/Stack2 \n");
    scanf("%d",&op);
    switch(op)
    {
        case 1:if(s->top1<0)
            {
                printf("Stack is empty\n");
                break;
            }
        else
            {
                el=s->a[s->top1];
                printf("Deleted element from stack1 is %d\t",el);
            }
    }
}

```

```

        s->top1--;
    }
    break;
case 2:if(s->top2==sz)
    {
        printf("Stack is empty\n");
        break;
    }
    else
    {
        el=s->a[s->top2];
        printf("Deleted element from stack2 is:%d\t",el);
        s->top2++;
    }
    break;
}
return el;
}
void print(st s)
{
    int i,op;
    printf("Which Stack do u want to print\n");
    scanf("%d",&op);
    switch(op)
    {
        case 1:if(s->top1<0)
            {
                printf("Stack is empty\n");
                break;
            }
            else
            {
                for(i=s->top1;i>=0;i--)
                    printf("%d\t",s->a[i]);
            }
            break;
        case 2:if(s->top2>sz-1)
            {
                printf("Stack is empty\n");
                break;
            }
            else
            {
                for(i=s->top2;i<sz;i++)
                    printf("%d\t",s->a[i]);
            }
            break;
    }
}
}

```

**Output:**

```
3.Print
4.Exit
Enter ur choice
3
Which Stack do u want to print
1
11
Two Way Stack Operations
1.Push
2.Pop
3.Print
4.Exit
Enter ur choice
3
Which Stack do u want to print
2
Stack is empty

Two Way Stack Operations
1.Push
2.Pop
3.Print
4.Exit
Enter ur choice
```

**22. Add two polynomials using linked list.**

```

#include<stdio.h>
#include<conio.h>
struct polynomial
{
    int exp;
    int coe;
    struct polynomial *next;
};
typedef struct polynomial * node;
typedef struct polynomial * poly;
poly create();
void sum(poly,poly,poly);
void display(poly);
void read(poly);
void sort(poly);
poly start,ptr,ptr1;
poly create()
{
    poly t;
    printf("\nEnter the elements of polynomial\n");
    t=(poly)malloc(sizeof(poly));
    t->next=NULL;
    start=t;
    if(start==NULL)
    {
        printf("Unable to create memory\n");
        exit();
    }
    return t;
}
void read(poly p)
{
    char ch;
    poly t;
    t=(poly)malloc(sizeof(poly));
    start->next=t;
    while(1)
    {
        printf("Enter coef and exponent\n");
        scanf("%d\n%d",&t->coe,&t->exp);
        if(t->exp==0)
        {
            p=t;
            t=(poly)malloc(sizeof(poly));
            t=NULL;
            p->next=t;
            break;
        }
    }
}

```

```

        printf("Do u want to enter more coef?(y/n)\n");
        fflush(stdin);
        scanf("%c",&ch);
        if(ch=='n')
        {
            p=t;
            t=(poly)malloc(sizeof(poly));
            t=NULL;
            p->next=t;
            break;
        }
        p=t;
        t=(poly)malloc(sizeof(poly));
        p->next=t;
    }
}

void display(poly p)
{
    int i=1;
    p=start->next;
    while(p!=NULL)
    {
        if(i!=1)
            printf("+");
        printf("%dx^%d",p->coe,p->exp);
        p=p->next;
        i++;
    }
}

void sort(poly p)
{
    int temp1,temp2;
    poly p1;
    p=start->next;

    for(;p->coe!=NULL;p=p->next)
        for(p1=p->next;p1->coe!=NULL;p1=p1->next)
        {
            if(p->exp<p1->exp)
            {
                temp1=p->coe;
                temp2=p->exp;
                p->coe=p1->coe;
                p->exp=p1->exp;
                p1->coe=temp1;
                p1->exp=temp2;
            }
        }
}
}

```

```

void sum(poly p1,poly p2,poly p3)
{
    poly t;
    t=(poly)malloc(sizeof(poly));
    ptr=p1->next;
    ptr1=p2->next;
    start->next=t;
    while(ptr!=NULL && ptr1!=NULL)
    {
        p3->next=t;
        if(ptr->exp>ptr1->exp)
            {
                t->coe=ptr->coe;
                t->exp=ptr->exp;
                ptr=ptr->next;
            }
        else if(ptr->exp<ptr1->exp)
            {
                t->coe=ptr1->coe;
                t->exp=ptr1->exp;
                ptr1=ptr1->next;
            }
        else
            {
                t->coe=ptr->coe+ptr1->coe;
                t->exp=ptr1->exp;
                ptr=ptr->next;
                ptr1=ptr1->next;
            }
        p3=t;
        t=(poly)malloc(sizeof(poly));
        p3->next=t;
    }
    if(ptr==NULL)
    {
        while(ptr1!=NULL)
        {
            t->coe=ptr1->coe;
            t->exp=ptr1->exp;
            ptr1=ptr1->next;
            p3=t;
            t=(poly)malloc(sizeof(poly));
            p3->next=t;
        }
    }
    else if(ptr1!=NULL)
    {
        while(ptr1!=NULL)
        {
            t->coe=ptr->coe;
            t->exp=ptr->exp;

```

```

        ptr=ptr->next;
        p3=t;
        t=(poly)malloc(sizeof(poly));
        p3->next=t;
    }
}
t=NULL;
p3->next=t;
}

void main()
{
    poly p1,p2,p3;
    clrscr();
    p1=create();
    read(p1);
    printf("First polynomial is :\n");
    sort(p1);
    display(p1);
    p2=create();
    read(p2);
    printf("\nSecond polynomial is :\n");
    sort(p2);
    display(p2);
    printf("\nAddition of two polynomials is:\n");
    p3=create();
    sum(p1,p2,p3);
    sort(p3);
    display(p3);
    getch();
}

```

**Output:**

```

10
0
First polynomial is :
2x^2+3x^1+10x^0
Enter the elements of polynomial
Enter coef and exponent
2
2
Do u want to enter more coef?(y/n)
y
Enter coef and exponent
4
1
Do u want to enter more coef?(y/n)
y
Enter coef and exponent
12
0
Second polynomial is :
2x^2+4x^1+12x^0
Addition of two polynomials is:
Enter the elements of polynomial
4x^2+7x^1+22x^0

```



**23. Multiply two polynomials using linked list.**

```

#include<stdio.h>
#include<conio.h>
struct polynomial
{
    int exp;
    int coe;
    struct polynomial *next;
};
typedef struct polynomial * node;
typedef struct polynomial * poly;
poly create();
void mul(poly,poly,poly);
void display(poly);
void read(poly);
void sort(poly);
void destroy(poly);
poly previous(poly);
poly start,ptr,ptr1,ptr2;
poly create()
{
    poly t;
    t=(poly)malloc(sizeof(poly));
    t->next=NULL;
    t->exp=0;
    t->coe=0;
    start=t;
    if(start==NULL)
    {
        printf("Unable to create memory\n");
        exit();
    }
    return t;
}
void read(poly p)
{
    char ch;
    poly t;
    t=(poly)malloc(sizeof(poly));
    start->next=t;
    while(1)
    {
        printf("\nEnter coef and exponent\n");
        scanf("%d\n%d",&t->coe,&t->exp);
        if(t->exp==0)
        {
            p=t;
            t=(poly)malloc(sizeof(poly));
            t=NULL;
        }
    }
}

```

```

        p->next=t;
        break;
    }
    printf("Do u want to enter more coef?(y/n)\n");
    fflush(stdin);
    scanf("%c",&ch);
    if(ch=='n')
    {
        p=t;
        t=(poly)malloc(sizeof(poly));
        t=NULL;
        p->next=t;
        break;
    }
    p=t;
    t=(poly)malloc(sizeof(poly));
    p->next=t;
}
}
void display(poly p)
{
    int i=1;
    p=start->next;
    while(p!=NULL)
    {
        if(i!=1)
            printf("+");
        printf("%dx^%d",p->coe,p->exp);
        p=p->next;
        i++;
    }
}
void sort(poly p)
{
    int temp1,temp2;
    poly p1,t;
    p=start->next;

    for(;p->coe!=NULL;p=p->next)
        for(p1=p->next;p1->coe!=NULL;p1=p1->next)
        {
            if(p->exp<p1->exp)
            {
                temp1=p->coe;
                temp2=p->exp;
                p->coe=p1->coe;
                p->exp=p1->exp;
                p1->coe=temp1;
                p1->exp=temp2;
            }
        }
}

```

```

        }
    }
void mul(poly p1,poly p2,poly p3)
{
    poly t,p,ptr3,ptr4,t1;
    int c,e;
    p=create();
    ptr=p1->next;
    ptr1=p2->next;
    t=(poly)malloc(sizeof(poly));
    while(ptr!=NULL)
    {
        ptr1=p2->next;
        while(ptr1!=NULL)
        {
            if(p!=NULL)
                p->next=t;
            t->coe=ptr->coe*ptr1->coe;
            t->exp=ptr->exp+ptr1->exp;
            p=t;
            t=(poly)malloc(sizeof(poly));
            p->next=t;

            ptr1=ptr1->next;
        }
        ptr=ptr->next;
    }
    ptr3=ptr4=start->next;
    p3=create();
    t1=(poly)malloc(sizeof(poly));
    while(ptr3!=t)
    {
        e=ptr3->exp;
        c=ptr3->coe;
        ptr4=ptr3->next;
        while(ptr4!=t)
        {
            if(ptr3->exp==ptr4->exp)
            {
                c=c+ptr4->coe;
                ptr4->coe=0;
            }
            ptr4=ptr4->next;
        }
        if(c!=0)
        {
            if(p3!=NULL)
                p3->next=t1;

```

```

        t1->coe=c;
        t1->exp=e;
        p3=t1;
        t1=(poly)malloc(sizeof(poly));
        p3->next=t1;
    }

    ptr3=ptr3->next;

}
t1=NULL;
p3->next=t1;
}
void main()
{
    poly p1,p2,p3;
    clrscr();
    p1=create();
    read(p1);
    printf("First polynomial is :\n");
    sort(p1);
    display(p1);
    p2=create();
    read(p2);
    printf("\nSecond polynomial is :\n");
    sort(p2);
    display(p2);
    printf("\nMultiplication of two polynomials is:\n");
    mul(p1,p2,p3);
    sort(p3);
    display(p3);

    getch();
}

```

**Output:**

```

Enter coef and exponent
4
0
First polynomial is :
2x^2+3x^1+4x^0
Enter coef and exponent
2
2
Do u want to enter more coef?(y/n)
y
Enter coef and exponent
4
1
Do u want to enter more coef?(y/n)
y
Enter coef and exponent
5
0
Second polynomial is :
2x^2+4x^1+5x^0
Multiplication of two polynomials is:
4x^4+14x^3+30x^2+31x^1+20x^0

```

**24. Construct BST and implement traversing techniques recursively.**

```

#include<stdio.h>
#include<conio.h>
struct btree
{
    int el;
    struct btree *l;
    struct btree *r;
};
typedef struct btree *bst;
void inorder(bst);
void preorder(bst);
void postorder(bst);
bst create(bst);
bst root;
bst create(bst a)
{
    char ch1,ch2;
    a=(bst)malloc(sizeof(bst));
    printf("Enter element\n");
    scanf("%d",&a->el);
    a->l=NULL;
    a->r=NULL;
    printf("Is there any left child of %d",a->el);
    fflush(stdin);
    scanf("%c",&ch1);
    if(ch1=='y' || ch1=='Y')
    {
        a->l=create(a->l);
    }
    printf("Is there any right child of %d ",a->el);
    fflush(stdin);
    scanf("%c",&ch2);
    if(ch2=='y' || ch2=='Y')
    {
        a->r=create(a->r);
    }
    return a;
}
void inorder(bst root)
{
    if(root!=NULL)
    {
        inorder(root->l);
        printf("%d\t",root->el);
        if(root->l==NULL)

```

```

        root->r=NULL;
        inorder(root->r);
    }
}
void preorder(bst root)
{
    if(root!=NULL)
    {
        printf("%d\t",root->el);
        if(root->l!=NULL)
            root->r=NULL;
        preorder(root->l);
        preorder(root->r);
    }
}
void postorder(bst root)
{
    if(root!=NULL)
    {
        postorder(root->l);
        if(root->l!=NULL)
            root->r=NULL;
        postorder(root->r);
        printf("%d\t",root->el);
    }
}
void main()
{
    int n,i,ch;
    clrscr();
    while(1)
    {

        printf("Binary search tree operations\n");
        printf("1.Inorder\n2.Preorder\n3.Postorder\n4.Exit\n");
        printf("Enter ur choice\n");
        scanf("%d",&ch);
        switch(ch)
        {
            case 1:root=create(root);
                    printf("Inorder Traversal\n");
                    inorder(root);
                    break;
            case 2:printf("Preorder Traversal\n");
                    preorder(root);
                    break;
            case 3:printf("Postorder Traversal\n");
                    postorder(root);
                    break;
            case 4:exit(0);
        }
    }
}

```

```
    }  
}
```

**Output:**

```
Is there any right child of 50 n  
Inorder Traversal  
10    20    25    40    50    Binary search tree operations  
1.Inorder  
2.Preorder  
3.Postorder  
4.Exit  
Enter ur choice  
2  
Preorder Traversal  
40    20    10    25    50    Binary search tree operations  
1.Inorder  
2.Preorder  
3.Postorder  
4.Exit  
Enter ur choice  
3  
Postorder Traversal  
10    25    20    50    40    Binary search tree operations  
1.Inorder  
2.Preorder  
3.Postorder  
4.Exit  
Enter ur choice
```

**25. Implement preorder traversal on BST non recursively.**

```

#include<stdio.h>
#include<conio.h>
struct btree
{
    int el;
    struct btree * l;
    struct btree * r;
};
typedef struct btree * bst;
struct stack
{
    int top;
    bst *a;
};
typedef struct stack * st;
void push(st,bst);
bst pop(st);
int isempty(st);
bst create(bst);
void preorder(bst);
bst create(bst a)
{
    char ch1,ch2;
    a=(bst)malloc(sizeof(bst));
    printf("Enter element\n");
    scanf("%d",&a->el);
    a->l=NULL;
    a->r=NULL;
    printf("Is there any left child of %d ", a->el);
    fflush(stdin);
    scanf("%c",&ch1);
    if(ch1=='y' || ch1=='Y')
    {
        a->l=create(a->l);
    }
    printf("Is there any right child of %d ",a->el);
    fflush(stdin);
    scanf("%c",&ch2);
    if(ch2=='y' || ch2=='Y')
    {
        a->r=create(a->r);
    }
    return a;
}
void push(st s,bst t)
{
    s->top++;
    s->a[s->top]=t;
}

```



```

}
bst pop(st s)
{
    bst t;
    t=s->a[s->top];
    s->top--;
    return t;
}
int isempty(st s)
{
    if(s->top==0)
        return 1;
    else
        return 0;
}
bst tops(st s)
{
    return s->a[s->top];
}
void preorder(bst t)
{
    bst t1=t;
    st s;
    push(s,t1);
    while(!isempty(s))
    {
        t1=pop(s);
        if(t1!=NULL)
        {
            printf("%d\t",t1->el);
            if(t1->l==NULL)
                t1->r=NULL;
            push(s,t1->r);
            push(s,t1->l);
        }
    }
}
}
void main()
{
    bst root;
    clrscr();
    root=create(root);
    printf("Pre Order Traversal\n");
    preorder(root);
    getch();
}

```

**Output:**

```
Enter element
40
Is there any left child of 40 y
Enter element
20
Is there any left child of 20 y
Enter element
10
Is there any left child of 10 n
Is there any right child of 10 n
Is there any right child of 20 y
Enter element
25
Is there any left child of 25 n
Is there any right child of 25 n
Is there any right child of 40 y
Enter element
50
Is there any left child of 50 n
Is there any right child of 50 n
Pre Order Traversal
40      20      10      25      50
```

**26. Implement postorder traversal on BST non recursively.**

```

#include<stdio.h>
#include<conio.h>
struct btree
{
    int el;
    struct btree * l;
    struct btree * r;
};
typedef struct btree * bst;
struct stack
{
    int top;
    bst *a;
};
typedef struct stack * st;
void push(st,bst);
bst pop(st);
int isempty(st);
bst create(bst);
void postorder(bst);
bst create(bst a)
{
    char ch1,ch2;
    a=(bst)malloc(sizeof(bst));
    printf("Enter element\n");
    scanf("%d",&a->el);
    a->l=NULL;
    a->r=NULL;
    printf("Is there any left child of %d ", a->el);
    fflush(stdin);
    scanf("%c",&ch1);
    if(ch1=='y' || ch1=='Y')
    {
        a->l=create(a->l);
    }
    printf("Is there any right child of %d ",a->el);
    fflush(stdin);
    scanf("%c",&ch2);
    if(ch2=='y' || ch2=='Y')
    {
        a->r=create(a->r);
    }
    return a;
}
void push(st s,bst t)
{
    s->top++;
    s->a[s->top]=t;
}

```

```

}
bst pop(st s)
{
    bst t;
    t=s->a[s->top];
    s->top--;
    return t;
}
int isempty(st s)
{
    if(s->top==0)
        return 1;
    else
        return 0;
}
bst tops(st s)
{
    return s->a[s->top];
}
void postorder(bst t)
{
    bst t1=t;
    st s;
    while(t1!=NULL)
    {
        push(s,t1);
        if(t1->l!=NULL)
            t1->r=NULL;
        if(t1->r!=NULL)
            push(s,t1->r);
        t1=t1->l;
    }
    while(!isempty(s))
    {
        t1=pop(s);
        printf("%d\t",t1->el);
    }
}
}
void main()
{
    bst root;
    clrscr();
    root=create(root);
    printf("Pre Order Traversal\n");
    postorder(root);
    getch();
}

```

**Output:**

```
Enter element
40
Is there any left child of 40 y
Enter element
20
Is there any left child of 20 y
Enter element
10
Is there any left child of 10 n
Is there any right child of 10 n
Is there any right child of 20 y
Enter element
25
Is there any left child of 25 n
Is there any right child of 25 n
Is there any right child of 40 y
Enter element
50
Is there any left child of 50 n
Is there any right child of 50 n
Pre Order Traversal
10    25    20    50    40
```

**27. Implement inorder traversal on BST non recursively.**

```

#include<stdio.h>
#include<conio.h>
struct btree
{
    int el;
    struct btree * l;
    struct btree * r;
};
typedef struct btree * bst;
struct stack
{
    int top;
    bst *a;
};
typedef struct stack * st;
void push(st,bst);
bst pop(st);
int isempty(st);
bst create(bst);
void inorder(bst);
bst create(bst a)
{
    char ch1,ch2;
    a=(bst)malloc(sizeof(bst));
    printf("Enter element\n");
    scanf("%d",&a->el);
    a->l=NULL;
    a->r=NULL;
    printf("Is there any left child of %d",a->el);
    fflush(stdin);
    scanf("%c",&ch1);
    if(ch1=='y' || ch1=='Y')
    {
        a->l=create(a->l);
    }
    printf("Is there any right child of %d ",a->el);
    fflush(stdin);
    scanf("%c",&ch2);
    if(ch2=='y' || ch2=='Y')
    {
        a->r=create(a->r);
    }
    return a;
}
void push(st s,bst t)
{
    s->top++;
    s->a[s->top]=t;
}

```

```

bst pop(st s)
{
    bst t;
    t=s->a[s->top];
    s->top--;
    return t;
}
int isempty(st s)
{
    if(s->top==0)
        return 1;
    else
        return 0;
}
void inorder(bst t)
{
    bst t1=t;
    st s;
    do
    {
        while(t1!=NULL)
        {
            push(s,t1);
            t1=t1->l;
        }
        if(!isempty(s))
        {
            t1=pop(s);
            printf("%d\t",t1->el);
            if(t1->l==NULL)
                t1->r=NULL;
            t1=t1->r;
        }
    }
    while(!isempty(s) || t1!=NULL);
}
void main()
{
    bst root;
    clrscr();
    root=create(root);
    inorder(root);
    getch();
}

```

**Output:**

```
Enter element
40
Is there any left child of 40y
Enter element
20
Is there any left child of 20y
Enter element
10
Is there any left child of 10n
Is there any right child of 10 n
Is there any right child of 20 y
Enter element
25
Is there any left child of 25n
Is there any right child of 25 n
Is there any right child of 40 y
Enter element
50
Is there any left child of 50n
Is there any right child of 50 n
10      20      25      40      50      =
```

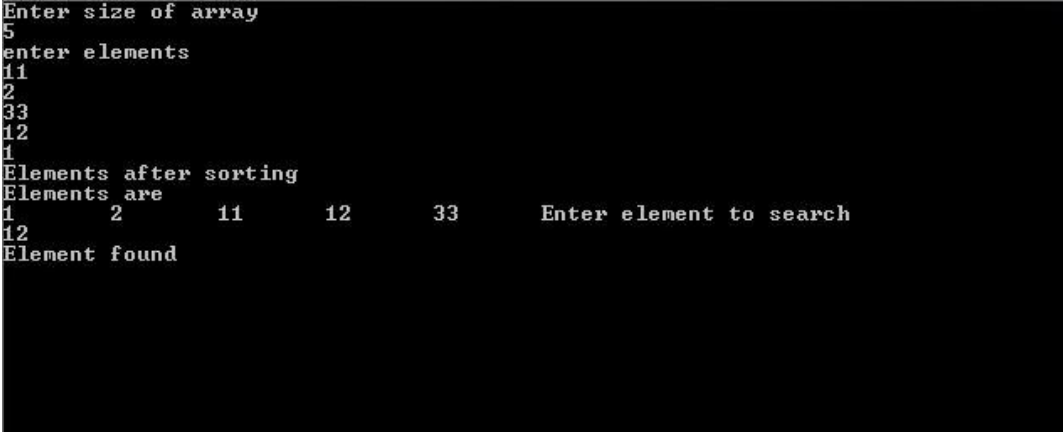


**28. Implement binary search technique recursively.**

```
#include<stdio.h>
#include<conio.h>
void binary(int [],int ,int);
void main()
{
    int a[10],i,n,j,el,temp;
    clrscr();
    printf("Enter size of array\n");
    scanf("%d",&n);
    printf("enter elements\n");
    for(i=0;i<n;i++)
    {
        scanf("%d",&a[i]);
    }
    printf("Elements after sorting \n");
    for(i=0;i<n;i++)
    {
        for(j=i+1;j<n;j++)
        {
            if(a[i]>a[j])
            {
                temp=a[i];
                a[i]=a[j];
                a[j]=temp;
            }
        }
    }
    printf("Elements are\n");
    for(i=0;i<n;i++)
    {
        printf("%d\t",a[i]);
    }
    printf("Enter element to search\n");
    scanf("%d",&el);
    binary(a,n,el);
    getch();
}
void binary(int a[],int n,int el)
{
    int l,h,mid,flag=0;
    l=0;
    h=n-1;
    while(l<h)
    {
        mid=(l+h)/2;
        if(el==a[mid])
        {
            flag=1;
        }
    }
}
```

```
        break;
    }
    else
    {
        if(e1>a[mid])
            l=mid+1;
        else
            h=mid-1;
    }
}
if(flag==1)
    printf("Element found\n");
else
    printf("Not found\n");
}
```

**Output:**



```
Enter size of array
5
enter elements
11
2
33
12
1
Elements after sorting
Elements are
1      2      11      12      33      Enter element to search
12
Element found
```

29. Program to implement graph traversals BFS and DFS.

```

#include<stdio.h>
#include<conio.h>
#define m 10
int adj[m][m];
typedef enum boolean{false,true} bool;
bool visited[m];
int n;
void create();
void display();
void dfs(int );
void bfs(int);
void create()
{
    int i,e,s,d;
    printf("Enter no of nodes");
    scanf("%d",&n);
    e=n*(n-1);
    for(i=0;i<=e;i++)
    {
        printf("Enter edges %d(0 0 to quit):",i);
        scanf("%d %d",&s,&d);
        if((s==0) && (d==0))
            break;
        if(s>n || d>n || s<=0 || d<=0)
        {
            printf("invalid edge\n");
            i--;
        }
        else
        {
            adj[s][d]=1;
        }
    }
}
void display()
{
    int i,j;
    for(i=1;i<=n;i++)
    {
        for(j=1;j<=n;j++)
            printf("%d\t", adj[i][j]);
        printf("\n");
    }
}

void dfs(int v)

```

```

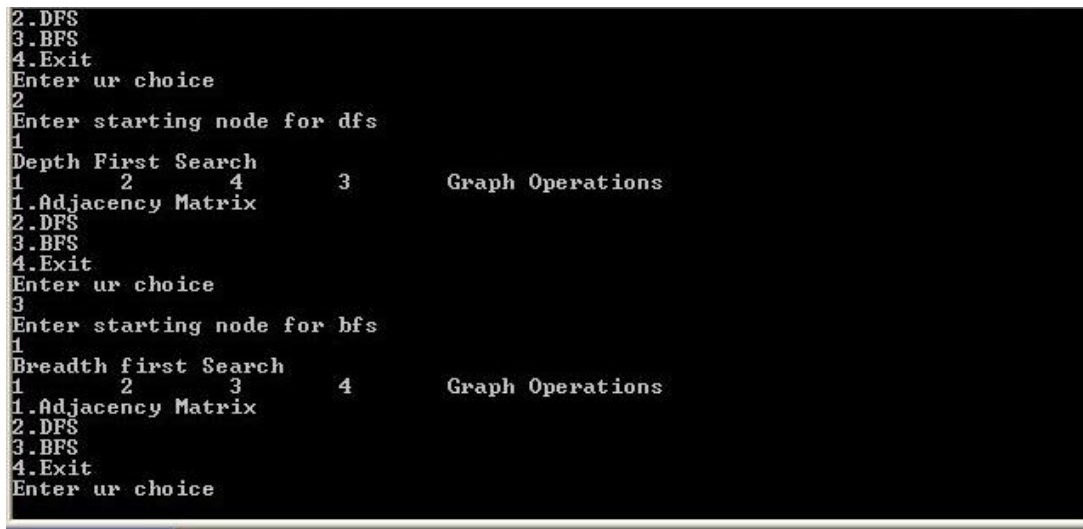
{
    int i;
    visited[v]=true;
    printf("%d\t",v);
    for(i=1;i<=n;i++)
        if(adj[v][i]==1 && visited[i]==false)
            dfs(i);
}
void bfs(int v)
{
    int i,front,rear;
    int que[20];
    front=rear=1;
    printf("%d\t",v);
    visited[v]=true;
    rear++;
    que[rear]=v;
    while(front<=rear)
    {
        v=que[front];
        front++;
        for(i=1;i<=n;i++)
        {
            if(adj[v][i]==1 && visited[i]==false)
            {
                printf("%d\t",i);
                visited[i]=true;
                rear++;
                que[rear]=i;
            }
        }
    }
}
void main()
{
    int i,v,choice;
    clrscr();
    create();
    while(1)
    {
        printf("Graph Operations\n");
        printf("1.Adjacency Matrix\n2.DFS\n3.BFS\n4.Exit\n");
        printf("Enter ur choice\n");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:printf("Adjacency Matrix\n");
                    display();
                    break;
            case 2:printf("Enter starting node for dfs\n");

```

```
scanf("%d",&v);
printf("Depth First Search\n");
for(i=1;i<=n;i++)
visited[i]=false;

dfs(v);
break;
case 3: printf("Enter starting node for bfs\n");
scanf("%d",&v);
printf("Breadth first Search\n");
for(i=1;i<=n;i++)
visited[i]=false;
bfs(v);
break;
case 4:exit();
}
}
}
```

**Output:**



```
2.DFS
3.BFS
4.Exit
Enter ur choice
2
Enter starting node for dfs
1
Depth First Search
1      2      4      3      Graph Operations
1.Adjacency Matrix
2.DFS
3.BFS
4.Exit
Enter ur choice
3
Enter starting node for bfs
1
Breadth first Search
1      2      3      4      Graph Operations
1.Adjacency Matrix
2.DFS
3.BFS
4.Exit
Enter ur choice
```

**30. Program to estimate the shortest path for a graph.**

```

#include<stdio.h>
#include<conio.h>
#define m 10
int adj[m][m],d[m][m];
typedef enum boolean{false,true} bool;
bool visited[m];
int n;
void create(int);
void display(int );
void shortestpath(int);
int min(int,int);
void create(int n)
{
    int i,e,s,d,w;
    e=n*(n-1);
    for(i=0;i<=e;i++)
    {
        printf("Enter edges %d(0 0 to quit):",i);
        scanf("%d %d",&s,&d);
        if((s==0) && (d==0))
            break;
        printf("enter the weight between two edges\n");
        scanf("%d",&w);
        if(s>n || d>n || s<=0 || d<=0)
        {
            printf("invalid edge\n");
            i--;
        }
        else
        {
            adj[s][d]=w;
        }
    }
}
void display(int n)
{
    int i,j;
    for(i=1;i<=n;i++)
    {
        for(j=1;j<=n;j++)
            printf("%d\t", adj[i][j]);
        printf("\n");
    }
}

```

```

void shortestpath(int n)
{
    int a,b,k;
    a=b=k=1;
    for(a=1;a<=n;a++)
    {
        for(b=1;b<=n;b++)
            d[a][b]=adj[a][b];
    }
    for(k=1;k<=n;k++)
    {
        for(a=1;a<=n;a++)
        {
            for(b=1;b<=n;b++)
            {
                d[a][b]=min(d[a][b],d[a][k]+d[k][b]);
            }
        }
    }
    for(a=1;a<=n;a++)
    {
        for(b=1;b<=n;b++)
            printf("%d\t",d[a][b]);
        printf("\n");
    }
}

int min(int a,int b)
{
    if(a<b)
        return a;
    else
        return b;
}

void main()
{
    int i,n,j,cost;
    int d[m][m];
    clrscr();
    printf("Enter no of nodes");
    scanf("%d",&n);
    create(n);
    printf("Adjacency Matrix\n");
    display(n);
    printf("Shortest path\n");
    shortestpath(n);
    getch();
}

```

**Output:**

```
enter the weight between two edges
11
Enter edges 2(0 0 to quit):2
3
enter the weight between two edges
2
Enter edges 3(0 0 to quit):3
1
enter the weight between two edges
3
Enter edges 4(0 0 to quit):2
1
enter the weight between two edges
6
Enter edges 5(0 0 to quit):0
0
Adjacency Matrix
0      4      11
6      0      2
3      0      0
Shortest path
0      4      6
5      0      2
3      0      0
```



