PROGRAM 1

Write a C++ program to sort the given list using selection sort technique.

```
#include<iostream.h>
#include<conio.h>
#include<iomanip.h>
int i,j,temp,n,a[15];
class select
{
public:
void getdata();
void sort();
void display();
};
void select::getdata()
{

cout<<"Enter the range"<<endl;
cin>>n;
cout<<"enter the element"<<endl;
for(i=0;i<n;i++)
cin>>a[i];
}
void select::sort()
{
int i,j,pos,temp;
for(i=0;i<=n-1;i++)
{
pos=i;
for(j=i+1;j<=n-1;j++)
if(a[j]<=a[pos])
{
pos=j;
}
temp=a[i];
a[i]=a[pos];
a[pos]=temp;
}
}
void select::display()
{
cout<<"Elements sorted are "<<endl;
for(i=0;i<n;i++)
cout<<a[i]<<endl;
}
void main()
{
```

```
select s;
clrscr();
s.getdata();
s.sort();
s.display();
getch();
}
```

OUTPUT:

```
enter the range
5
enter the element
12 34 76 45 0
elements in sorted order are
0
12
34
45
76
```

PROGRAM 2

Write a C++ program to sort the given list using insertion sort technique.

```
#include<iostream.h>
#include<conio.h>
#include<iomanip.h>
int i,j,temp,n,a[15];
class insert
{
public:
void getdata();
void sort();
void display();
};
void insert::getdata()
{
cout<<"Enter the range"<<endl;
cin>>n;
cout<<"Enter the elements"<<endl;
for(i=0;i<n;i++)
cin>>a[i];
}
void insert::sort()
{
for(i=1;i<=n-1;i++)
{
j=i;
temp=a[i];
while(j>0&&(a[j-1]>temp))
{
a[j]=a[j-1];
j--;
}
a[j]=temp;
}
}
void insert::display()
{
cout<<"Elements sorted are"<<endl;
for(i=0;i<n;i++)
cout<<a[i]<<endl;
}
void main()
{
insert i;
clrscr();
i.getdata();
```

```
i.sort();
i.display();
getch();
}
```

OUTPUT:

```
Enter the range
5
Enter the element
12 99 37 88 26
Sorted elements are
12
26
37
88
99
```

PROGRAM 3

Write a C++ program to sort the given list using bubble sort technique.

```
#include<iostream.h>
#include<conio.h>
#include<iomanip.h>
int i,j,temp,n,a[15];
class bubble
{
public:
void getdata();
void sort();
void display();
};
void bubble::getdata()
{
cout<<"Enter the range"<<endl;
cin>>n;
cout<<"Enter the element"<<endl;
for(i=0;i<n;i++)
cin>>a[i];
}
void bubble::sort()
{
int i,j,temp;
for(i=1;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;
}
}
}
void bubble::display()
{
cout<<"Elements sorted are"<<endl;
for(i=0;i<n;i++)
cout<<a[i]<<endl;
}
void main()
{
bubble b;
b.getdata();
b.sort();
```

```
b.display();
clrscr();
getch();
}
```

OUTPUT:

```
enter the range
4
enter the element
89
77
51
40
sorted elements are
40
51
77
89
```

PROGRAM 4

Write a C++ program to search an element using linear search technique.

```
#include<iostream.h>
#include<conio.h>
#include<iomanip.h>
void main()
{
int a[20],i,j,n,pos,key,flag=0;
clrscr();
cout<<"Enter the range"<<endl;
cin>>n;
cout<<"Enter the elements"<<endl;
for(i=0;i<n;i++)
cin>>a[i];
cout<<"Enter the elements to search"<<endl;
cin>>key;
for(i=0;i<n;i++)
if(key==a[i])
{
flag=1;
pos=i;
break;
}
if(flag==1)
cout<<"The element found in position "<<pos<<endl;
else
cout<<"The element is not found in the list"<<endl;
getch();
}
```

OUTPUT:

```
enter the range
5
enter the elements
23 45 67 89 2
enter the elements to search
2
the element is found in position 4
_
```

```
enter the range
5
enter the elements
23 45 67 89 2
enter the elements to search
88
the element is not found in the  list
```

PROGRAM 5

Write a C++ program to search an element using binary search technique.

```
#include<iostream.h>
#include<conio.h>
#include<iomanip.h>
void getdata();
int a[15],n;
void sort();
void bsearch();
void getdata()
{
int i;
cout<<"Enter the range"<<endl;
cin>>n;
cout<<"Enter the elements"<<endl;
for(i=0;i<n;i++)
cin>>a[i];
}
void sort()
{
int i,j,temp;
for(i=1;i<=n-1;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;
}
cout<<"Elements in the sorted order are "<<endl;
for(i=0;i<n;i++)
cout<<setw(5)<<a[i]<<endl;
}
}
void bsearch()
{
int key,mid,flag=0,lb=0,ub=n-1;
cout<<"Enter the elements to search"<<endl;
cin>>key;
lb=0;
ub=n-1;
while(lb<=ub)
{
mid=(lb+ub)/2;
if(a[mid]==key)
```

```
{
flag=1;
break;
}
if(key>a[mid])
lb=mid+1;
else
ub=mid-1;
}
if(flag==1)
{
cout<<"Elements found at "<<mid+1<<"position"<<endl;
}
else
{
cout<<"Elements not found "<<endl;
}
}
void main()
{
clrscr();
getdata();
sort();
bsearch();
getch();
}
```

OUTPUT:

```
enter the range:
5
enter the elements:
99 0 12 34 56
elements in the sorted order are
    0
   12
   34
   56
   99
enter the element to be searched
45
  element not found
-
```

```
enter the range:
5
enter the elements:
67 0 99 8 12
elements in the sorted order are
    0
    8
   12
   67
   99
enter the element to be searched
12
element is found at 3 position
```

PROGRAM 6

Write a C++ program for Towers of Hanoi.

```
#include<iostream.h>
#include<conio.h>
#include<iomanip.h>
void towers(int,char,char,char);
void main()
{
int n;
clrscr();
cout<<"Enter the number"<<endl;
cin>>n;
towers(n,'A','B','C');
getch();
}
void towers(int n,char start,char aux,char last)
{
if(n==1)
{
cout<<"Move disk 1 from "<<start<<"to "<<last<<endl;
return;
}
towers(n-1,start,last,aux);
cout<<"Move disk "<<n<<" from "<<start<<"to "<<last<<endl;
towers(n-1,aux,start,last);
}
```

OUTPUT:

```
Enter the number
3
Move disk 1 from A to C
Move disk 2 from A to B
Move disk 1 from C to B
Move disk 3 from A to C
Move disk 1 from B to A
Move disk 2 from B to C
Move disk 1 from A to C
_
```

PROGRAM 7

Write a C++ program to implement dynamic array. Also find smallest and largest element.

```
#include<iostream.h>
#include<conio.h>
#include<stdio.h>
#include<stdlib.h>
void main()
{
int i,n,*arr,max,min;
clrscr();
cout<<"Enter the size of the array: "<<endl;
cin>>n;
arr=(int*)calloc(n,sizeof(int));
cout<<"Enter the array elements: "<<endl;
for(i=0;i<n;i++)
{
cin>>arr[i];
}
max=arr[0];
for(i=0;i<n;i++)
{
if(max>arr[i])
{
max=arr[i];
}
}
min=arr[0];
for(i=0;i<n;i++)
{
if(min<arr[i])
{
min=arr[i];
}
}
cout<<"Max and min numbers in the array are: "<<max<<" and "<<min;
getch();
}
```

OUTPUT:

```
enter the size of the array:
5
enter the array elements:
12 88 0 34 99
max and min numbers in array are:99 and 0_
```

PROGRAM 1

Write a C++ program to sort the given list using the merge sort technique.

```
#include<iostream.h>
#include<conio.h>
#include<iomanip.h>
void mergesort(int a[],int,int);
void mergearray(int a[],int,int,int);
int a[20],n;
void main()
{
int i,low,high,mid;
clrscr();
cout<<"Enter the range: "<<endl;
cin>>n;
cout<<"Enter the elements: "<<endl;
for(i=0;i<n;i++)
cin>>a[i];
low=0;
high=n-1;
mergesort(a,low,high);
cout<<"The sorted array is: "<<endl;
for(i=0;i<n;i++)
cout<<a[i]<<endl;
getch();
}
void mergesort(int a[],int low,int high)
{
int mid;
if(low<high)
{
mid=(low+high)/2;
mergesort(a,low,mid);
mergesort(a,mid+1,high);
mergearray(a,low,mid,high);
}
}
void mergearray(int a[],int low,int mid,int high)
{
int c[20],i,j,k;
i=low;
j=mid+1;
k=low;
while((i<=mid)&&(j<=high))
if(a[i]<a[j])
c[k++]=a[i++];
else
```

```
c[k++]=a[j++];
while(i<=mid)
c[k++]=a[i++];
while(j<=high)
c[k++]=a[j++];
for(i=low;i<=high;i++)
a[i]=c[i];
}
```

OUTPUT:

```
Enter the range:
4
Enter the elements:
67
56
98
23
The sorted array is:
23
56
67
98
```

PROGRAM 2

Write a C++ program to sort the given list using the quick sort technique.

```
#include<iostream.h>
#include<conio.h>
#include<iomanip.h>
int a[50],n;
int partition(int a[],int,int);
void quicksort(int a[],int,int);
void main()
{
clrscr();
int i;
cout<<"Enter the range: "<<endl;
cin>>n;
cout<<"Enter the elements: "<<endl;
for(i=0;i<n;i++)
cin>>a[i];
quicksort(a,0,n-1);
cout<<"Element in sorted order are: "<<endl;
for(i=0;i<n;i++)
cout<<a[i]<<endl;
getch();
}
void quicksort(int a[],int low,int high)
{
int loc;
if(low<high)
{
loc=partition(a,low,high);
quicksort(a,low,loc-1);
quicksort(a,loc+1,high);
}
}
int partition(int a[],int low,int high)
{
int pivot,i,j,temp;
pivot=a[low];
i=low+1;
j=high;
while(i<high&&pivot>=a[i])
i++;
while(pivot<a[j])
j--;
if(i<j)
{
temp=a[i];
```

```
a[i]=a[j];
a[j]=temp;
}
else
{
temp=a[low];
a[low]=a[j];
a[j]=temp;
}
return(j);
}
```

OUTPUT:

```
Enter the range:
4
Enter the elements:
98
45
77
21
Element in sorted order are:
21
45
77
98
```

PROGRAM 3

Write a C++ program to implement stack operations using arrays.

```
#include<iostream.h>
#include<conio.h>
#include<iomanip.h>
#include<stdlib.h>
#define size 5
class stack
{
private:
int s[size],n,i,top;
public:
stack();
void push();
void pop();
 void display();
};
stack::stack()
{
top=-1;
}
void stack::push()
{
if(top>=size-1)
{
cout<<"STACK IS OVERFLOW"<<endl;
return;
}
top++;
cout<<"Enter the element: "<<endl;
cin>>n;
s[top]=n;
}
void stack::pop()
{
if(top==-1)
{
cout<<"STACK IS UNDERFLOW"<<endl;
return;
}
n=s[top];
top--;
cout<<"THE DELETED ELEMENT IS: "<<n<<endl;
}
void stack::display()
{
```

```
if(top==-1)
{
cout<<"STACK IS EMPTY"<<endl;
return;
}
cout<<"STACK ELEMENTS ARE: "<<endl;
for(i=top;i>=0;i--)
cout<<s[i]<<endl;
}
void main()
{
stack s;
clrscr();
int ch;
do
{
cout<<"STACK MENU"<<endl;
cout<<"1.Push"<<endl;
cout<<"2.Pop"<<endl;
cout<<"3.Display"<<endl;
cout<<"4.Exit"<<endl;
cout<<"Enter the choice: "<<endl;
cin>>ch;
switch(ch)
{
case 1:
{
s.push();
break;
}
case 2:
{
s.pop();
break;
}
case 3:
{
s.display();
break;
}
case 4:
{
exit(0);
}
default:
{
cout<<"INVALID INPUT"<<endl;
}
```

```
}
}
while(ch<=4);
getch();
}
```

OUTPUT:

```
STACK MENU
1.Push
2.Pop
3.Display
4.Exit
Enter the choice:
2
STACK IS UNDERFLOW
STACK MENU
1.Push
2.Pop
3.Display
4.Exit
Enter the choice:
3
STACK IS EMPTY
STACK MENU
1.Push
2.Pop
3.Display
Enter the choice:
1
Enter the element:
24
```

```
STACK MENU
1.Push
2.Pop
3.Display
4.Exit
Enter the choice:
1
STACK IS OVERFLOW

STACK MENU
1.Push
2.Pop
3.Display
4.Exit
Enter the choice:
3
STACK ELEMENTS ARE:
24
STACK MENU
1.Push
2.Pop
3.Display
4.Exit
Enter the choice:
2
THE DELETED ELEMENT IS: 24

STACK MENU
1.Push
2.Pop
3.Display
4.Exit
Enter the choice:
5
INVALID INPUT
```

PROGRAM 4

Write a C++ program to implement queue operations using arrays.

```
#include<iostream.h>
#include<conio.h>
#include<iomanip.h>
#include<stdlib.h>
#define SIZE 5
class queue
{
private:
int q[SIZE],front,rear;
public:
queue();
void qinsert();
void qdelete();
void qdisplay();
};
queue::queue()
{
front=-1;
rear=-1;
}
void queue::qinsert()
{
int num;
if(rear==SIZE-1)
{
cout<<"Queue Overflow"<<endl;
return;
}
rear++;
cout<<"Enter the element to be inserted"<<endl;
cin>>num;
q[rear]=num;
if(front==-1)
{
front=0;
return;
}
}
void queue::qdelete()
{
int num;
if(front==-1)
{
cout<<"Queue Underflow"<<endl;
```

```
return;
}
num=q[front];
cout<<"The deleted element is: "<<endl<<num<<endl;
if(front==rear)
{
front=-1;
rear=-1;
}
else
{
front++;
return;
}
}
void queue::qdisplay()
{
int I;
if(front==-1&&rear==-1)
{
cout<<"Queue is empty"<<endl;
return;
}
cout<<"The elemets in the queue are: "<<endl;
for(i=front;i<=rear;i++)
cout<<q[i]<<"\t";
}
void main()
{
queue q;
int ch;
clrscr();
do
{
cout<<endl<<"******MENU******"<<endl;
cout<<"1.Insert"<<endl;
cout<<"2.Delete"<<endl;
cout<<"3.Display"<<endl;
cout<<"4.Exit"<<endl;
cout<<"Enter the choice: "<<endl;
cin>>ch;
switch(ch)
{
case 1:
{
q.qinsert();
break;
}
```

```
case 2:
{
q.qdelete();
break;
}
case 3:
{
q.qdisplay();
break;
}
case 4:
{
exit(0);
}
default:
{
cout<<"Invalid Input"<<endl;
}
}
}
while(ch<=4);
getch();
}
```

OUTPUT:

```
*******MENU*******
1.Insert
2.Delete
3.Display
4.Exit
Enter the choice:
2
Queue Underflow

*******MENU*******
1.Insert
2.Delete
3.Display
4.Exit
Enter the choice:
3
Queue is empty
```

```
*******MENU*******
1.Insert
2.Delete
3.Display
4.Exit
Enter the choice:
1

Enter the element to be inserted
25

*******MENU*******
1.Insert
2.Delete
3.Display
4.Exit
Enter the choice:
1
Queue Overflow

*******MENU*******
1.Insert
2.Delete
3.Display
4.Exit
Enter the choice:
3
The elemets in the queue are:
25
```

```
*******MENU*******
1.Insert
2.Delete
3.Display
4.Exit
Enter the choice:
2
The deleted element is:
25


*******MENU*******
1.Insert
2.Delete
3.Display
4.Exit
Enter the choice:
5
Invalid Input
```

PROGRAM 5

Write a C++ program to evaluate postflix expression.

```
#include<iostream.h>
#include<conio.h>
#include<string.h>
#include<stdlib.h>
#include<ctype.h>
#include<iomanip.h>
int s[10],top=-1;
void push(int);
int pop();
void push(int x)
{
s[++top]=x;
}
int pop()
{
return(s[top--]);
}
void main()
{
clrscr();
char expr[20],ch;
int op1,op2,n;
cout<<"Enter the expression: "<<endl;
cin>>expr;
n=strlen(expr);
for(int i=0;i<n;i++)
{
ch=expr[i];
if(isdigit(ch))
push(ch-'0');
else
{
op1=pop();
op2=pop();
switch(ch)
{
case '+':
{
push(op2+op1);
break;
}
case '-':
{
push(op2-op1);
```

```
break;
}
case '*':
{
push(op2*op1);
break;
}
case '/':
{
push(op2/op1);
break;
}
}
}
}
cout<<"The result is = "<<s[top]<<endl;
getch();
}
```

OUTPUT:

```
enter the expression
426*+4-
the result is=12
-
```

PROGRAM 6

Write a C++ program to implement a circular queue using an array.

```
#include<iostream.h>
#include<conio.h>
#include<iomanip.h>
#include<stdlib.h>
#define max 5
class cqueue{
private:int cq[max],front,rear;
public:
void cqinsert();
void cqdelete();
void cqdisplay();
cqueue()
{
front=-1;
rear=-1;
}
};
void cqueue::cqinsert()
{
 int num;
if((rear+1)%max==front||(front==0&&rear==max-1))
{
 cout<<"circular queue overflow"<<endl;
 return;
 }
else
{
cout<<"enter the element to be inserted"<<endl;
cin>>num;
if(front==-1)
front=0;
 rear=(rear+1)%max;
 cq[rear]=num;
}
}
void cqueue::cqdelete()
{
int num;
if(front==-1)
{
 cout<<"circular queue underflow"<<endl;
 return;
}
num=cq[front];
```

```
 if(front==rear)
{
front=-1 ;
rear=-1;
}

else
{
front=(front+1)%max;
}
cout<<"the deleted element is:"<<num<<endl;
}
void cqueue::cqdisplay()
{
int i=front;
if(front==-1)
{
cout<<"circular queue is empty"<<endl;
return;
}
else
{
cout<<"the elements in circular queue are:"<<endl;
while(i<=rear)
{
cout<<cq[i]<<"\t"<<endl;
i=i+1%max;
}
}
}
void main()
{
cqueue c;
clrscr();
int ch;
do
{
cout<<"***MENU***"<<endl;
cout<<"1.insert"<<endl;
cout<<"2.delete"<<endl;
cout<<"3.display"<<endl;
cout<<"4.exit"<<endl;
cout<<"enter your choice"<<endl;
cin>>ch;
switch(ch)
{
case 1:
{
```

```
c.cqinsert();
break;
}
case 2:
{
c.cqdelete();
break;
}
case 3:
{
c.cqdisplay();
break;
}
case 4:
{
exit(0);
}
default:
{
cout<<"invalid choice"<<endl;
}
}
}
while(ch<=4);
getch();

}
```

OUTPUT:

```
***MENU***
1.Insert
2.Delete
3.Display
4.Exit
Enter your choice:
2
circular queue underflow
```

```
Enter your choice:
1
enter the element to be inserted
34


***MENU***
1.Insert
2.Delete
3.Display
4.Exit
Enter your choice:
1
circular queue overflow

***MENU***
1.Insert
2.Delete
3.Display
4.Exit
Enter your choice:
3
the elements in circular queue are:
23      45

***MENU***
1.Insert
2.Delete
3.Display
4.Exit
Enter your choice:
2
the deleted element is:23

Enter your choice:

2
the deleted element is:45
```

```
***MENU***
1.Insert
2.Delete
3.Display
4.Exit
Enter your choice:
3
circular queue is empty


***MENU***
1.Insert
2.Delete
3.Display
4.Exit
Enter your choice:
5
Invalid choice
```

PROGRAM 7

Write a C++ program to implement stack operations using a linked list.

```
#include<iostream.h>
#include<conio.h>
#include<iomanip.h>
#include<stdlib.h>
#include<process.h>
struct list
{
int data;
struct list*link;
};
typedef struct list*NODE;
NODE getnode();
NODE insert_front(NODE,int);
NODE delete_front(NODE);
void display(NODE);
void main()
{
NODE first=NULL;
int ch,num;
clrscr();
do
{
cout<<"Stack using linked list are: "<<endl;
cout<<"1.Insert"<<endl;
cout<<"2.Delete"<<endl;
cout<<"3.Display"<<endl;
cout<<"4.Exit"<<endl;
cout<<"Enter your choice: "<<endl;
cin>>ch;
switch(ch)
{
case 1:
{
cout<<"Enter the elements to be inserted: "<<endl;
cin>>num;
first=insert_front(first,num);
break;
}
case 2:
{
first=delete_front(first);
break;
}
case 3:
```

```
{
display(first);
break;
}
case 4:
{
exit(0);
}
default:
{
cout<<"Invalid Choice"<<endl;
}
}
}
while(ch<=4);
getch();
}
NODE getnode()
{
NODE x;
x=(NODE)malloc(sizeof(NODE));
return(x);
}
NODE insert_front(NODE first,int num)
{

NODE temp;
temp=getnode();
temp->data=num;
temp->link=first;
return temp;
}
NODE delete_front(NODE first)
{
NODE temp;
if(first==NULL)
{
cout<<"List is empty"<<endl;
return(first);
}
temp=first;
temp=temp->link;
cout<<"The deleted element is: "<<first->data<<endl;
free(first);
return(temp);
}
void display(NODE first)
{
```

```
NODE temp;
if(first==NULL)
{
cout<<"List is empty";
return;
}
cout<<"Element in the list are "<<endl;
temp=first;
cout<<"The element in the list are: ";
while(temp!=NULL)
{
cout<<temp->data<<"-->";
temp=temp->link;
}
cout<<"NULL"<<endl;
}
```

OUTPUT:

```
Stack using linked list are:
1.Insert
2.Delete
3.Display
4.Exit
Enter your choice:
2
List is empty

Stack using linked list are:
1.Insert
2.Delete
3.Display
4.Exit
Enter your choice:
3
List is empty
```

```
Stack using linked list are:
1.Insert
2.Delete
3.Display
4.Exit
Enter your choice:
1
Enter the elements to be inserted:
22

Stack using linked list are:
1.Insert
2.Delete
3.Display
4.Exit
Enter your choice:
3
Element in the list are
The element in the list are: 22-->NULL

Stack using linked list are:
1.Insert
2.Delete
3.Display
4.Exit
Enter your choice:
2
The deleted element is: 22

Stack using linked list are:
1.Insert
2.Delete
3.Display
4.Exit
Enter your choice:
5
Invalid Choice
```

PROGRAM 8

Write a C++ program to implement queue operations using a linked list.

```
#include<iostream.h>
#include<conio.h>
#include<stdlib.h>
#include<process.h>
#include<iomanip.h>
struct list
{
int data;
struct list * link;
};
typedef struct list * NODE;
NODE getnode();
void insertRear();
void deleteFront();
void displayNodes();
NODE front=NULL,rear=NULL;
void main()
{
int ch;
clrscr();
do{
cout<<endl<<"Queue using linked list are: "<<endl;
cout<<"1.Insert"<<endl;
cout<<"2.delete"<<endl;
cout<<"3.Display"<<endl;
cout<<"4.Exit"<<endl;
cout<<"Enter the choice: "<<endl;
cin>>ch;
switch(ch)
{
case 1:
{
insertRear();
break;
}
case 2:
{
deleteFront();
break;
}
case 3:
displayNodes();
break;
case 4:
```

```
exit(0);
default:
cout<<"Invalid choice";
}
}
while(ch<=4);
getch();
}
NODE getnode()
{
NODE x;
x=(NODE)malloc(sizeof(NODE));
return x;
}
void insertRear()
{
NODE temp;
int num;
temp=getnode();
cout<<"Enter the number to be inserted: ";
cin>>num;
temp->data=num;
temp->link=NULL;
if(front==NULL)
{
front=rear=temp;
}
else
{
rear->link=temp;
rear=temp;
}
}
void deleteFront()
{
NODE temp;
if(front==NULL)
{
cout<<"Linked list is empty";
}
else
{
temp=front;
cout<<"The delete element is: "<<front->data;
if(front==rear)
{
front=rear=NULL;
}
```

```
else
{
front=front->link;
}
free(temp);
}
}
void displayNodes()
{
NODE temp;
if(front==NULL)
{
cout<<"Linked list is empty";
}
else
{
temp=front;
cout<<"The element is the list are: ";
while(temp!=NULL)
{
cout<<temp->data<<"->";
temp=temp->link;
}
cout<<"NULL";
}
}
```

OUTPUT:

```
Queue using linked list are:
1.Insert
2.delete
3.Display
4.Exit
Enter the choice:
2
Linked list is empty
Queue using linked list are:
1.Insert
2.delete
3.Display
4.Exit
Enter the choice:
3
Linked list is empty
```

```
Queue using linked list are:
1.Insert
2.delete
3.Display
4.Exit
Enter the choice:
1
Enter the number to be inserted: 55

Queue using linked list are:
1.Insert
2.delete
3.Display
4.Exit
Enter the choice:
3
The element is the list are: 55->NULL

Queue using linked list are:
1.Insert
2.delete
3.Display
4.Exit
Enter the choice:
2
The delete element is: 55

Queue using linked list are:
1.Insert
2.delete
3.Display
4.Exit
Enter the choice:
5
Invalid choice
```