```
#include<iostream>
#include<stdlib.h>
using namespace std;
template<class T>class Node{
public:
   T info;
   Node *next,*prev;
   Node(T data){
      info=data;
      next=NULL;
      prev=NULL;
   }

};
template<class T>class DLL{
   Node<T> *head,*tail;
   int count;
public:
   DLL(){
       head=tail=NULL;
       count=0;
   }
   void InsertAtBeg(T data){
       count++;
       Node<T> *newNode=new
Node<T>(data);
       if(head==NULL)
           head=tail=newNode;
       else
          {
           newNode->next=head;
           head->prev=newNode;
           head=newNode;
       }
   }
   void InsertAtEnd(T data){
       count++;
       Node<T> *newNode=new
Node<T>(data);
       if(tail==NULL)
           head=tail=newNode;
       else
          {
           tail->next=newNode;
           newNode->prev=tail;
           tail=newNode;
       }
   }

   T DelFromBeg(){
       count--;
       if(head==NULL) //case 1 when linked
list is empty
           throw "Linked list is empty ";
       else if(head==tail)    //case 2 when
linked list contains single node
       {
           T data=head->info;
           delete head;
           head=tail=NULL;
           return data;
       }
       else    //case 3 when linked list
contains more than one node
       {
           T data=head->info;
           Node<T> *temp=head;
           head=head->next;
           delete temp;
           head->prev=NULL;
           return data;
       }
```

```
  }
  T Del_From_End(){
      count--;
      if(head==NULL) //case 1 when linked
list is empty
          throw "Linked list is empty ";
      else if(head==tail)    //case 2 when
linked list contains single node
      {
          T data=tail->info;
          delete tail;
          head=tail=NULL;
          return data;
      }
      else    //case 3 when linked list
contains more than one node
      {
          T data=tail->info;
```

```
        Node<T> *Current=tail;
        tail=tail->prev;
        delete Current;
        tail->next=NULL;
        return data;
    }
}
void Display(){
    Node<T> *current=head;
    cout<<"Linked List : ";
    if(current!=NULL)

        while(current!=NULL)
        {
            cout<<current->info<<"  ";
            current=current->next;
        }
    else
```

```
            cout<<"Empty";
    cout<<endl;
}
void  Count(){
    cout<<"\nNo of nodes are "<<count;
}
bool  Search_Value(T val){
    if(head==NULL)
        throw  "Linked List is empty ";
    else
       {
        Node<T> *temp=head;
        while(temp!=NULL)
        {
            if(temp->info==val)
            {
                return  true;
                break;
            }
            temp=temp->next;
        }
        return false;
    }
}
void Reverse(){
    if(head==NULL)
        throw "Linked list is empty. ";
    else if(head==tail)
        cout<<"Nothing can be done. ";
    else{
        Node<T> *temp;
        Node<T> *current=head;
        while(current!=NULL)
        {
            temp=current->next;
            current->next=current->prev;
        current->prev=temp;
        current=current->prev;
      }
      temp=head;
      head=tail;
      tail=temp;
  }
}
void InsertAtPos(T data,int pos){
  Node<T> *temp=new Node<T>(data);
  if(pos<=count && pos>0)
  {
      if(head==NULL)
          throw "Linked List is empty ";
      else
          {
          Node<T> *current=head;
          for(int i=1;i<pos-1;i++)
      {
        current=current->next;
      }
      temp->next=current->next;
      current->next=temp;
      temp->prev=current;
    }
}
else
        cout<<"Error";
count++;

}

T DelValue(T value){
if(head==NULL) //case 1 when linked
list is empty
    throw "Linked list is empty ";
else if(head==tail &&
value==head->info){
    T data=tail->info;
    delete tail;
    head=tail=NULL;
    return data;
}
else{
    Node<T> *current=head;
    Node<T> *temp=head->next;
    Node<T> *temp2;
    while(value!=current->info)
    {
        current=current->next;
        temp=temp->next;
    }
    T data=current->info;
    temp->prev=current->prev;
    temp2=current->prev;
    delete current;
    temp2->next=temp;
    return data;
  }

}
void menu(){
  T ch;
  cout<<" MENU FOR DOUBLY
LINKED LIST ";
  cout<<"\n1.Insert node at the
beginning. ";
  cout<<"\n2.Insert node at the end. ";
  cout<<"\n3.Display Linked list.";
  cout<<"\n4.Delete node from the
beginning. ";
```

```
        cout<<"\n5.Delete node from the
End. ";
        cout<<"\n6.No. of Nodes. ";
        cout<<"\n7.Search Value. ";
        cout<<"\n8.Reverse of Linked list. ";
        cout<<"\n9.Insert node at given
position. ";
        cout<<"\n10.Delete a particular node.
";
        cout<<"\n11.Go back to menu ";
        choice();
    }
    void choice(){
        T value,n,K;
        int p,ch;
        bool k;
        cout<<"\nEnter your choice : ";
        cin>>ch;
```

```
      char c='Y';
      switch(ch) {
         case 1: cout<<"Enter the data to be
inserted : ";
               cin>>value;
               InsertAtBeg(value);
               break;
          case 2:    cout<<"Enter the data to
be inserted ";
               cin>>value;
               InsertAtEnd(value);
               break;
          case 3: Display();
               break;
          case 4: try{
                   K=DelFromBeg();
                   cout<<"Value Deleted
"<<K<<endl;
```

```
            }
            catch(const char *msg)
            {
                cout<<msg<<endl;
            }

            break;
        case 5:try{
                K=Del_From_End();
                cout<<"Value Deleted :
"<<K<<endl;
            }
            catch(const char *msg){
                cout<<msg<<endl;
            }
            break;
        case 6:Count();
            break;
        case 7:try{
                cout<<"Enter no. to be
searched ";
                cin>>n;
                k=Search_Value(n);
                if(k)
                    cout<<"Value Found ";
                else
                            cout<<"Value not
found ";
            }
            catch(const char *msg){
                cout<<msg<<endl;
            }
            break;

        case 8: try{
                Reverse();
```

```
    }
    catch(const char *msg)
    {
        cout<<msg<<endl;
    }
    break;
case 9:
    cout<<"Enter data : ";
    cin>>value;
    cout<<"Enter position : ";
    cin>>p;
    InsertAtPos(value,p);
    break;

case 10: try{
        cout<<"Enter position : ";
        cin>>p;
        K=DelValue(p);
```

```
                    cout<<"Value Deleted
"<<K<<endl;
                }
                catch(const char *msg){
                   cout<<msg<<endl;
                }
                break;
            case 11:menu();
            default:cout<<"Wrong Input";
        }
        cout<<"\nDo you want to
continue(Y/N) : ";
        cin>>c;
        if(c=='y' || c=='Y')
            choice();
        else {
            cout<<"\nExiting this program!!.
"<<endl;
```

```
            }
        }
    };
    int main(){
        DLL<int> ob;
        DLL<float> ob2;
        ob.menu();
        ob2.menu();
        return 0;
    }
```