**2. Write a template based C++ program that uses functions to perform the following:**

**a) Create a doubly linked list of elements.**

**b) Delete a given element from the above doubly linked list.**

**c) Display the contents of the above list after deletion**.

#include<iostream>

using namespace std;

template <class T>

class node

{

public:

node \*prev;

T data;

node \*next;

};

template <class T>

class dll

{

node<T> \*start;

public:

dll()

{

start=NULL;

}

void insertatbegin(T data);

void insertatend(T data);

void insertatpos(T value, int position);

void deleteelement(T value);

void display();

void count();

void reverse();

};

template<class T>void dll<T>::insertatbegin(T value)//10 20

{

node<T> \*ptr;

ptr=new node<T>;

ptr->data=value;//20

if (start == NULL)//NULL==NULL, 1001==NULL(F)

{

ptr->prev=NULL;

ptr->next=NULL;

start=ptr;

return;

}

ptr->next=start;

start->prev=ptr;

ptr->prev=NULL;

start=ptr;

}

template<class T>void dll<T>::insertatend(T value)//30

{

node<T> \*ptr,\*lp;

ptr=new node<T>;

ptr->data=value;//30

if (start == NULL)

{

ptr->prev=NULL;

ptr->next=NULL;

start=ptr;

return;

}

lp=start;

while(lp->next!=NULL)

{

lp=lp->next;

}

ptr->prev=lp;

lp->next=ptr;

ptr->next=NULL;

}

template<class T>void dll<T>::insertatpos(T value, int pos)

{

node<T> \*ptr,\*temp,\*ptemp;

int i=1;

ptr=new node<T>;

ptr->data=value;

if (start == NULL || pos == 1)

{

if(!start)

{

start= ptr;

ptr->prev=ptr->next=NULL;

return;

}

ptr->next = start;

start->prev=ptr;

ptr->prev=NULL;

start = ptr;

return;

}

temp = start;

while (temp)

{

if (pos == i + 1)

{

temp->next=ptr;

ptr->next=ptemp;

ptemp->prev=ptr;

ptr->prev=temp;

break;

}

i++;

temp = temp->next;

ptemp=temp->next;

}

}

template<class T>void dll<T>::deleteelement(T value)

{

node<T> \*pt,\*t,\*nt;

int res=0;

if(start==NULL)

{

cout<<"dll is empty\n";

return;

}

if(value==start->data)

{

t=start;

start=start->next;

delete t;

if(start!=NULL)

{

start->prev=NULL;

}

return;

}

pt=start;

t=start->next;

while(t!=NULL&&t->data!=value)

{

pt=t;

t=t->next;

}

pt->next=t->next;

if(t->next!=NULL)

{

nt=t->next;

nt->prev=pt;

res=1;

delete t;

}

if(res==0)

{

cout<<"the element is not found\n";

}

}

template<class T>void dll<T>::display()

{

node<T> \*temp;

if (start == NULL)

{

cout<<"List empty,nothing to display"<<endl;

return;

}

temp = start;

cout<<"The Doubly Link List is :"<<endl;

while (temp != NULL)

{

cout<<temp->data<<" ";

temp = temp->next;

}

cout<<endl;

}

template<class T>void dll<T>::count()

{

node<T> \*q = start;

int cnt = 0;

while (q != NULL)

{

q = q->next;

cnt++;

}

cout<<"Number of elements are: "<<cnt<<endl;

}

template<class T>void dll<T>::reverse()

{

node<T> \*p1, \*p2;

p1 = start;

p2 = p1->next;

p1->next = NULL;

p1->prev = p2;

while (p2 != NULL)

{

p2->prev = p2->next;

p2->next = p1;

p1 = p2;

p2 = p2->prev;

}

start = p1;

}

int main()

{

int choice, element, position;

dll<int> ob;

cout<<endl<<"----------------------------"<<endl;

cout<<endl<<"Operations on Doubly linked list"<<endl;

cout<<endl<<"----------------------------"<<endl;

cout<<"1.Insertion at begining"<<endl;

cout<<"2.Insertion at end"<<endl;

cout<<"3.Insertion after position"<<endl;

cout<<"4.Delete"<<endl;

cout<<"5.Display"<<endl;

cout<<"6.count\n";

cout<<"7.Reverse"<<endl;

cout<<"8.Quit"<<endl;

while(1)

{

cout<<"Enter your choice : \n";

cin>>choice;

switch ( choice )

{

case 1:

cout<<"Enter the element:\n ";

cin>>element;

ob.insertatbegin(element);

break;

case 2:

cout<<"Enter the element: ";

cin>>element;

ob.insertatend(element);

break;

case 3:

cout<<"Enter the element: ";

cin>>element;

cout<<"Insert Element after postion: ";

cin>>position;

ob.insertatpos(element, position);

break;

case 4:

cout<<"Enter the element for deletion: ";

cin>>element;

ob.deleteelement(element);

break;

case 5:

ob.display();

break;

case 6:

ob.count();

break;

case 7:

ob.reverse();

break;

case 8:

exit(0);

default:

cout<<"Wrong choice"<<endl;

}

}

return 0;

}