lab 4 work:

#include"header.h"

using namespace std;

int main()

{

Node<int> \*a, \*b, \*c, \*d, \*e;

a = new Node<int>(200);

b = new Node<int>(30);

c = new Node<int>(40);

d = new Node<int>(45);

e = new Node<int>(450);

DList<int> \*list;

list = new DList<int>();

list->Insert(0 , a);

list->Insert(a , b);

list->Insert(b , c);

list->Insert(a , d);

list->Insert(b, e);

list->printList();

list->Delete(a);

cout<<"\nAfter deleting first node"<<endl;

/\*list->printList();

list->Swap(b,c);

cout<<"\nAfterswaping the two nodes"<<endl;

list->printList();

\*/

system("pause");

return 0;

}

header:

#ifndef DLIST\_H

#define DLIST\_H

template<class T>

class Node

{

private:

T data;

Node \*next ;

Node \*previous ;

public:

// constructor

Node(T pdata)

{

data=pdata;

next=NULL;

previous=NULL;

}

//sets the data in the Node

void setData(T pVal)

{

data=pVal;

}

// returns the T data in the Node

T getData()

{

return data;

}

// returns the link to the next node

Node\* getNext()

{

return next;

}

// sets the link to the next node

void setNext(Node\* x)

{

next=x;

}

// returns the link to the previous node

Node\* getPrevious()

{

return previous;

}

// sets the link to the previous node

void setPrevious(Node\* x)

{

previous=x;

}

};

template<class T>

class DList

{

private:

Node<T> \*first ;

public:

DList()

{

first=NULL;

}

//Inserts the node pNew after the node pBefore

// if the list is empty, it makes pNew the first node of the list

void Insert(Node<T>\* pBefore, Node<T>\* pNew)

{

if(first)

{

pNew->setNext(first);

first->setPrevious(pNew);

first=pNew;

pNew->setPrevious(NULL);

}

else if(first==NULL)

{

first=pNew;

pNew->setNext(NULL);

pNew->setPrevious(NULL);

}

else

{

pNew->setNext(pBefore->getNext());

pNew->setPrevious(pBefore);

pBefore->setNext(pNew);

}

}

void Delete(Node<T>\* pToBeDeleted)

{

if (pToBeDeleted==first->getNext())

{

first->setNext(pToBeDeleted->getNext());

}

/\*else if (pToBeDeleted !=NULL && pToBeDeleted!=first->getNext())

{

}\*/

delete pToBeDeleted;

}

void printList()

{

Node<T> \*temp;

temp=first;

do

{

cout<<temp->getData()<<" ";

temp=temp->getNext();

}

while ( temp!=NULL );

}

//Swaps the two nodes in the list.

void Swap(Node<T>\* Node1, Node<T>\* Node2);

};

#endif