POSTLAB CODE:

Header file:

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 DCList

{

private:

Node<T> \*first;

public:

DCList() { 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 (pBefore == NULL)

{

first = pNew;

pNew->setNext(NULL);

pNew->setPrevious(NULL);

}

else if ((pBefore == first && pBefore->getNext() != NULL) && pBefore->getNext()!=first)

{

Node<T>\*temp1 = first;

while (temp1->getNext() != first)

{

temp1 = temp1->getNext();

}

temp1->setNext(pNew);

pNew->setNext(first);

pNew->setPrevious(temp1);

first->setPrevious(pNew);

first = pNew;

}

else if (pBefore->getPrevious() == first && pBefore->getNext()==first)

{

pNew->setPrevious(pBefore);

pNew->setNext(first);

first->setPrevious(pNew);

pBefore->setNext(pNew);

}

else if (pBefore == first && pBefore->getNext() == NULL)

{

pBefore->setNext(pNew);

pNew->setNext(pBefore);

pNew->setPrevious(pBefore);

pBefore->setPrevious(pNew);

}

else

{

Node<T>\*temp2;

temp2 = pBefore->getNext();

pNew->setNext(temp2);

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

pNew->setPrevious(pNew);

pBefore->setNext(pNew);

}

}

//Deletes the node pToBeDeleted

void Delete(Node<T>\* pToBeDeleted){

if (pToBeDeleted == first)

{

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

first->getNext()->setPrevious(first->getPrevious());

first = first->getNext();

delete pToBeDeleted;

}

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

{

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

first->setPrevious(pToBeDeleted->getPrevious());

delete pToBeDeleted;

}

else if (pToBeDeleted == NULL)

{

cout << "No node to delete there ! " << endl;

}

else

{

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

pToBeDeleted->getNext()->setPrevious(pToBeDeleted->getPrevious());

delete pToBeDeleted;

}

}

//prints the contents of the list

void printList(){

Node<T> \*temp;

temp = first;

do

{

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

temp = temp->getNext();

} while (temp->getNext() != first);

}

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

if (Node1 == first && Node2->getNext() == first)

{

Node<T>\*temp1 = first;

Node2->getPrevious()->setNext(Node1);

Node1->setNext(Node2);

}

else if (Node1->getNext()==Node2)

{

Node1->getPrevious()->setNext(Node2);

Node2->getNext()->setPrevious(Node1);

Node2->setNext(Node1);

Node1->setPrevious(Node2);

}

}

};

MAIN FILE:

#include<iostream>

#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);

DCList<int> \*list;

list = new DCList<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 deletion : " << endl;

list->printList();

list->Swap(b,c);

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

list->printList();

system("pause");

return 0;

}