//

// main.cpp

// Lab4-LinkedList

//

// Created by Stephanie Marin Velasquez on 3/29/16.

// Copyright © 2016 Stephanie Marin Velasquez. All rights reserved.

//

#include <iostream>

using namespace std;

class LinkedList{

private:

struct node {

int data;

node \* next;

};

node \*head;

int size;

public:

LinkedList(): head(0), size(0){};

~LinkedList();

bool SortedIsEmpty()const; // return true if the list is empty... find out if list is empty or not

int SortedGetLenght(); // returnthe lenght of the list;

bool SortedFind(int k, int& x);

int SortedSearch(int key); //return the position of the element which is same as key(note: use the most efficient method)

void SortedInsert(int x); // insert x to the right position in the sorted linked list

void SortedDelete(int k, int &x, bool & success); // delete the kth element

void Print()const;

friend ostream& operator << ( ostream& out, const LinkedList &x); /// output the number

friend istream& operator >> (istream& in, LinkedList& num); // input the number

// Part (1) implement the aboved singly linked list allowing the duplicae element.

// Part (2) implement he above singly linked list without allowing the duplicate element.

//node \*getAt(int k)const;

};

LinkedList num;

LinkedList :: ~LinkedList()

{

node \* tmp;

while (head) {

tmp=head;

head=head->next;

delete tmp;

}

}

bool LinkedList:: SortedIsEmpty()const //// EMPTY

{

if (head==0) {return true;}return false;

}

int LinkedList:: SortedGetLenght()

{

return size;

}

bool LinkedList:: SortedFind(int k, int &x) ///FIND

{

if (head ==0 || k>size || k<= 0) {return false;}

int cnt= 1;

if(size==cnt) {x=head->data;}

node \*temp=head;

while(cnt <= size)

{

if (cnt == k)

{

x= temp->data;

return true;

}

else

{ temp=temp->next; cnt ++;}

}

return false;

}

int LinkedList:: SortedSearch(int key) ///SEARCH

{

int position = 1;

//if (head && head->next == 0 && head->data==key) return position;

//else

//{

node \*temp = head;

while (temp)

{

if (temp->data == key)return position;

else

{

temp = temp->next;

position++;

}

}

//if (position > size)

return -1;

//}

}

void LinkedList:: SortedInsert(int x) //INSERT

{

node \*newnode= new node;

newnode->data = x;

newnode->next=NULL;

if (size==0)

{

head=newnode;

}

else

{

node \*pre = head, \*cur = head;

if (newnode->data < cur->data )

{

newnode->next = cur;

head = newnode;

}

else

{

while (cur && cur->data <= newnode->data) {

pre = cur;

cur = cur->next; }

if (newnode->data == pre->data)

{

delete newnode;

return;

}

else

{

newnode->next = cur;

pre->next = newnode;

}

}

}

size++;

}

void LinkedList:: SortedDelete(int k, int &x, bool &success) //DELETE

{

//Case 1 - List is empty or the kth element does not exist.

if (k > size || k < 1){success = false; return;}

node \*pre=head,\* temp = head;

int counter = 1;

if (k == 1)

{//if kth element is the first in the list, head will be head->next

head = head->next;

x = temp->data;

delete temp;

success = true;

temp = 0;

}

else

{

while (temp)

{

if (counter == k)

{//if the element to delete was finding

x = temp->data;

pre->next = temp->next;

delete temp;

success = true;

temp = 0;

}

else

{

counter++;

pre = temp;

temp = temp->next;

}

}

}

size--;//lenght decrease

}

void LinkedList:: Print()const

{

if (head==0)

{

cout<<" \n\n This list does not have any element.\n";

}

else

{

cout<< "\t The numbers in this list are:\n" <<"\t: ";

node \*temp=head;

while (temp)

{

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

temp=temp->next;

}

cout<< " :";

}cout<<endl;

}

int main()

{

int opt;

bool answer = true;

while (answer)

{

cout<< "Enter the digit for the operation"<<endl;

cout<< "\t 1: Is Empty \n " <<"\t 2: Get Lenght \n "<< "\t 3: Find \n ";

cout<< "\t 4: Search \n" << "\t 5: Insert \n " << "\t 6: Delete \n"<< "\t 7: Print"<< "\t 8: Exit "<<endl<<endl;

cin>> opt;

switch (opt) {

case 1:

if ( num.SortedIsEmpty())

cout<< " The list is empty\n ";

else

cout<< "This list has element";

break;

case 2:

if (num.SortedGetLenght())

{

cout<< "The lenght of this list is: \n "<< ": "<< num.SortedGetLenght();

}

else

cout<< "sorry, this list is empty.";

break;

case 3:

int k, xValue;

cout<< "What's the position of the element you want to find? \t ";

cin>>k;

if (num.SortedFind(k, xValue))

{

cout<< "The element ont the " << k << "position is "<< xValue<<endl;

}

else

cout<< "The value of the" << k << "position does not extist. ";

break;

case 4:

int ElemS, place; // ElemS = element to search

cout<< "What element you want to search: ";

cin>> ElemS;

place= num.SortedSearch(ElemS);

if (place == -1)

{

cout<< " the element is not in the list. \n";

}

else

cout<< "The element is "<< ElemS<< "is at the place number" << place<< endl;

break;

case 5:

int number;

cout<<"Insert the number: ";

cin>> number;

num.SortedInsert(number); cout<<endl;

break;

case 6:

if (num.SortedGetLenght()==0)

{ // Deleting a number when there's nothing in there

cout<< " You have no elements to delete." <<endl;

}

else

{

bool success;

int elementDelete, place;

cout<<"Input the place kth to delete: ";

cin>> place;

num.SortedDelete(place, elementDelete, success);

if (success)

{

cout<< "The element deleted in the "<<place <<"th place has been deleted";

}

else

cout<< " The element "<< elementDelete << "does not exist in this list";

}

break;

case 7:

cout<< "This is the list: \n\n";

num.Print();cout<<endl;

case 8:

cout<< " Program finished\n ";

answer =false;

break;

default:

cout<< " Option does not exist";

break;

}

} /// end of Do

}