// Austin Faulkner: May 24, 2020

// A class template for holding a linked list.

// The node type is also a class template.

#ifndef LINKEDLIST\_H

#define LINKEDLIST\_H

#include <iostream>

namespace My\_Templated\_LL

{

template <class T>

class ListNode

{

public:

T data;

ListNode<T> \*next;

// Constructor

ListNode (T nodeValue)

{ data = nodeValue;

next = nullptr;}

};

template <class T>

class LinkedList

{

private:

ListNode<T> \*head; // List head pointer

public:

// Constructor

LinkedList()

{ head = nullptr; }

// Destructor

~LinkedList();

// Linked list operations

int FindListLength(ListNode<T>\* head);

bool IsSortedUp(ListNode<T>\* head);

void InsertAsHead(ListNode<T>\*& head, T data);

void InsertAsTail(ListNode<T>\*& head, T data);

void InsertSortedUp(ListNode<T>\*& head, T data, double value);

//----------------------------------------------------------------------------------

bool DeleteFirstTargetNode(ListNode<T>\*& head, T target);

(FIX - OPTION (F))

void DeleteNode(T target);

(OR: FIX - OPTION (F))

//----------------------------------------------------------------------------------

void ShowAll(std::ostream& outs, ListNode<T>\* head);

void SortLinkedList(ListNode<T>\* head);

void FindMinMax(ListNode<T>\* head, double& min, double& max);

double FindAverage(ListNode<T>\* head);

void PromoteTarget(ListNode<T>\*& head, T target);

// Still need to append if entry not in LL already

void ListClear(ListNode<T>\*& head, int noMsg = 0);

//----------------------------------------------------------------------------------

void ReadStudentRecords(std::ifstream& fin, std::istream& ins, T data);

// FIX - OPTION (M)

//----------------------------------------------------------------------------------

void PrintStudentRecords(std::ofstream& fout, ListNode<T>\* head);

};

}

#include "Templated\_LL.template"

#endif