Pair Programming 8 Turn In

Name:Matthew Krahel Username: \_c1020a12

Partner name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Partner username: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_ I certify that my partner worked with me on this assignment.

SCORE: \_\_\_\_\_\_\_\_\_\_\_\_ (to be filled in by instructor)

8a (5 points)

Graphical user interface

Description automatically generated with medium confidence

Graphical user interface, application

Description automatically generated

**#ifndef NODE\_H**

**#define NODE\_H**

**#include "book.h"**

**class Node {**

**private:**

**Book book;**

**Node\* nextPtr;**

**Node\* prevPtr;**

**public:**

**Node();**

**Node( const Book& );**

**Node( const Node& );**

**virtual ~Node() {}**

**void setBook( const Book& );**

**Book getBook() const { return book; }**

**// LinkedList is a friend**

**friend class LinkedList;**

**};**

**#endif**

**//Node.cpp**

**#include "node.h"**

**Node::Node( ) { // Default constructor**

**nextPtr = NULL;**

**prevPtr = NULL;**

**}**

**Node::Node( const Book& newBook) { // add in book object**

**nextPtr = NULL;**

**prevPtr = NULL;**

**book = newBook;**

**}**

**Node::Node( const Node& newNode ) { // copy constructor**

**nextPtr = NULL;**

**prevPtr = NULL;**

**book = Book( newNode.book );**

**}**

**void Node::setBook( const Book& newBook ) {**

**book = newBook;**

**}**

**#ifndef LINKEDLIST\_H**

**#define LINKEDLIST\_H**

**#include "book.h"**

**#include "node.h"**

**class LinkedList {**

**public:**

**LinkedList() : headPtr( NULL), tailPtr( NULL ), count( 0 ) {}**

**LinkedList( const LinkedList& );**

**void insertNode( Node\* ); // at head**

**void traverseQueue() const;**

**void traverseStack() const;**

**Node\* findNode( const Book& ) const;**

**void deleteNode( Node\* );**

**Node\* dequeueNode(); // from tail**

**Node\* popNode(); // from head**

**virtual ~LinkedList() { deleteList(); }**

**void deleteList();**

**bool isEmpty() const { return headPtr == NULL; }**

**LinkedList& operator=( const LinkedList& );**

**private:**

**Node\* headPtr;**

**Node\* tailPtr;**

**int count;**

**};**

**#endif**

**//linkedlist.cpp**

**#include "linkedlist.h"**

**#include "node.h"**

**#include <iostream>**

**#include <cstdlib>**

**LinkedList::LinkedList( const LinkedList& newList ){**

**Node\* thisPtr;**

**Node\* curPtr;**

**if( newList.headPtr != NULL ){**

**curPtr = newList.headPtr;**

**headPtr = new Node( \*curPtr );**

**thisPtr = headPtr;**

**curPtr = curPtr->nextPtr;**

**while(curPtr != NULL){**

**thisPtr->nextPtr = new Node( \*curPtr );**

**thisPtr->nextPtr->prevPtr = thisPtr;**

**thisPtr = thisPtr->nextPtr;**

**curPtr = curPtr->nextPtr;**

**}**

**}**

**tailPtr = thisPtr;**

**count = newList.count;**

**}**

**void LinkedList::insertNode( Node\* nPtr ){**

**count++;**

**if ( headPtr == NULL )**

**{**

**headPtr = nPtr;**

**tailPtr = nPtr;**

**}**

**else**

**{**

**nPtr->nextPtr = headPtr;**

**headPtr->prevPtr = nPtr;**

**headPtr = nPtr;**

**}**

**}**

**void LinkedList::traverseQueue() const{**

**Node\* curPtr = tailPtr;**

**while ( curPtr != NULL )**

**{**

**cout << curPtr->book;**

**curPtr = curPtr->prevPtr;**

**}**

**}**

**void LinkedList::traverseStack() const{**

**Node\* curPtr = headPtr;**

**while ( curPtr != NULL )**

**{**

**cout << curPtr->book;**

**curPtr = curPtr->nextPtr;**

**}**

**}**

**Node\* LinkedList::findNode( const Book& key ) const{**

**for( Node\* curPtr = headPtr; curPtr != NULL; curPtr = curPtr->nextPtr ){**

**if( curPtr->book == key){**

**return curPtr;**

**}**

**}**

**return NULL;**

**}**

**void LinkedList::deleteNode( Node\* nodePtr ){**

**if( nodePtr == headPtr ){**

**headPtr = nodePtr->nextPtr;**

**if( headPtr != NULL ){**

**headPtr->prevPtr = NULL;**

**} else {**

**tailPtr = NULL;**

**}**

**}**

**else if( nodePtr == tailPtr ){**

**tailPtr = nodePtr->prevPtr;**

**tailPtr->nextPtr = NULL;**

**} else {**

**nodePtr->prevPtr->nextPtr = nodePtr->nextPtr;**

**nodePtr->nextPtr->prevPtr = nodePtr->prevPtr;**

**}**

**count--;**

**free( nodePtr );**

**}**

**Node\* LinkedList::dequeueNode(){**

**count--;**

**Node\* nodePtr;**

**nodePtr = tailPtr;**

**tailPtr = tailPtr->prevPtr;**

**if ( count != 0 )**

**{**

**tailPtr->nextPtr = NULL;**

**}**

**else**

**{**

**headPtr = NULL;**

**}**

**nodePtr->prevPtr = NULL;**

**nodePtr->nextPtr = NULL;**

**return nodePtr;**

**}**

**Node\* LinkedList::popNode(){**

**Node\* nodePtr;**

**count--;**

**nodePtr = headPtr;**

**headPtr = headPtr->nextPtr;**

**if ( count != 0 )**

**{**

**headPtr->prevPtr = NULL;**

**}**

**else**

**{**

**tailPtr = NULL;**

**}**

**nodePtr->prevPtr = NULL;**

**nodePtr->nextPtr = NULL;**

**return nodePtr;**

**}**

**void LinkedList::deleteList(){**

**if(this != NULL && count != 0){**

**Node\* curPtr;**

**curPtr = headPtr;**

**while(curPtr != NULL ){**

**headPtr = curPtr->nextPtr;**

**free(curPtr);**

**curPtr = headPtr;**

**}**

**tailPtr = NULL;**

**count = 0;**

**}**

**}**

**LinkedList& LinkedList::operator=( const LinkedList& newList){**

**deleteList();**

**Node\* thisPtr;**

**if( newList.headPtr != NULL ){**

**Node\* curPtr = newList.headPtr;**

**headPtr = new Node( \*curPtr );**

**thisPtr = headPtr;**

**curPtr = curPtr->nextPtr;**

**while( curPtr != NULL ){**

**thisPtr->nextPtr = new Node( \*curPtr );**

**thisPtr->nextPtr->prevPtr = thisPtr;**

**thisPtr = thisPtr->nextPtr;**

**curPtr = curPtr->nextPtr;**

**}**

**}**

**tailPtr = thisPtr;**

**count = newList.count;**

**return \*this;**

**}**

8b (5 points)

Graphical user interface

Description automatically generated with medium confidence

**#ifndef BOOK\_H**

**#define BOOK\_H**

**#include <iostream>**

**#include <string>**

**using namespace std;**

**class Book{**

**private:**

**string ISBN;**

**string title;**

**string author;**

**string publisher;**

**int year;**

**int edition;**

**public:**

**//constructors**

**Book(); //Default constructor**

**Book( string, string, string, string, int, int ); //constructs more info**

**//copy constructor**

**Book( const Book& );**

**//destructor**

**virtual ~Book(){}**

**//set/get for private data**

**void setISBN( string );**

**string getISBN() const { return ISBN; }**

**void setTitle( string );**

**string getTitle() const{ return title; }**

**void setAuthor( string );**

**string getAuthor() const{ return author; }**

**void setPublisher( string );**

**string getPublisher() const { return publisher; }**

**void setYear( int );**

**int getYear() const { return year; }**

**void setEdition( int );**

**int getEdition() const { return edition; }**

**void setBook( const Book& );**

**// overloaded assignment operator**

**Book& operator=( const Book& );**

**// overloaded << and >>**

**friend ostream& operator<<( ostream&, const Book& );**

**friend istream& operator>>( istream&, Book& );**

**// overloaded ==**

**friend bool operator==( const Book&, const Book& );**

**// make Node class a friend**

**friend class Node;**

**ostream& write( ostream& ) const;**

**istream& read( istream& );**

**};**

**#endif**

**//book.cpp**

**#include "book.h"**

**#include <string>**

**#include <stdio.h>**

**#include <string.h>**

**#include <stdlib.h>**

**Book::Book(){**

**ISBN = "";**

**title = "";**

**author = "";**

**publisher = "";**

**year = 1;**

**edition = 1;**

**}**

**Book::Book( string i, string t, string a, string p, int y, int e){**

**setISBN( i );**

**setTitle( t );**

**setAuthor( a );**

**setPublisher( p );**

**setYear( y );**

**setEdition( e );**

**}**

**Book::Book( const Book& newBook){**

**setBook( newBook );**

**}**

**void Book::setISBN( string newISBN){**

**ISBN = newISBN;**

**}**

**void Book::setTitle( string newTitle ){**

**title = newTitle;**

**}**

**void Book::setAuthor( string newAuthor ){**

**author = newAuthor;**

**}**

**void Book::setPublisher( string newPublisher ){**

**publisher = newPublisher;**

**}**

**void Book::setYear( int newYear ){**

**if( newYear > year ){**

**year = newYear;**

**}**

**}**

**void Book::setEdition( int newEdition ){**

**if( newEdition > edition ){**

**edition = newEdition;**

**}**

**}**

**void Book::setBook( const Book& newBook ){**

**setISBN( newBook.ISBN );**

**setTitle( newBook.title );**

**setAuthor( newBook.author );**

**setPublisher( newBook.publisher );**

**setYear( newBook.year );**

**setEdition( newBook.edition );**

**}**

**Book& Book::operator=( const Book& b1 ){**

**ISBN = b1.ISBN;**

**title = b1.title;**

**author = b1.author;**

**publisher = b1.publisher;**

**year = b1.year;**

**edition = b1.edition;**

**return( \*this );**

**}**

**ostream& operator<<( ostream& outstream, const Book& b ){**

**outstream << b.ISBN << " " << b.title << endl << b.author << endl << b.publisher << " " << b.year << " " << b.edition << endl;**

**return outstream;**

**}**

**istream& operator>>( istream& instream, Book& b ){**

**string tempString;**

**getline(instream, b.ISBN);**

**getline(instream, b.title);**

**getline(instream, b.author);**

**getline(instream, b.publisher);**

**instream >> b.year >> b.edition;**

**return instream;**

**}**

**bool operator==( const Book& b1, const Book& b2 ){**

**////bool returnValue;**

**////cout << "|" << b1.title << "| " << "|" << b2.title << "| " << endl;**

**////cout << "|" << b1.ISBN << "| " << "|" << b2.ISBN << "| " << endl;**

**////cout << "|" << b1.author << "| " << "|" << b2.author << "| " << endl;**

**////cout << "|" << b1.publisher << "| " << "|" << b2.publisher << "| " << endl;**

**////cout << "|" << b1.year << "| " << "|" << b2.year << "| " << endl;**

**////cout << "|" << b1.edition << "| " << "|" << b2.edition << "| " << endl;**

**if( b1.ISBN == b2.ISBN && b1.title == b2.title ){**

**return true;**

**}**

**////if( b1.ISBN == b2.ISBN && b1.title == b2.title && b1.author == b2.author && b1.publisher == b2.publisher && b1.year == b2.year && b1.edition == b2.edition ){**

**////return true;**

**////}**

**return false;**

**////cout << "return value : " << returnValue << endl;**

**}**

**ostream& Book::write( ostream& outBin ) const {**

**struct book\_t {**

**char ISBN[128];**

**char title[128];**

**char author[128];**

**char publisher[128];**

**int year;**

**int edition;**

**} b;**

**strcpy( b.ISBN, ISBN.c\_str() );**

**strcpy( b.title, title.c\_str() );**

**strcpy( b.author, author.c\_str() );**

**strcpy( b.publisher, publisher.c\_str() );**

**b.year = year;**

**b.edition = edition;**

**outBin.write( (char\*)&b, sizeof(book\_t) );**

**return outBin;**

**}**

**istream& Book::read( istream& input ){**

**int i = 0;**

**struct book\_t {**

**char ISBN[128];**

**char title[128];**

**char author[128];**

**char publisher[128];**

**int year;**

**int edition;**

**} b;**

**input.read((char\*)&b, sizeof(Book));**

**while(i < 4){**

**i++;**

**input.read((char\*)&b, sizeof(Book));**

**}**

**}**