CISC220 Lab 3

(80 pts, due next Sat night, midnight)

Part A: Classes/pointers(35 pts)

Write a header file, and an accompanying .cpp file for a class for a book. The book class should include:

1. (2 pts)Fields for the author's first name, last name, an array of 10 integers for ratings (on the heap – just because…), a field for the name of the book, and a field for the year in which it was published.
2. (2 pts) Two constructors
   a. The first constructor takes as input the first name, the last name, the name of the book, the year published, and then creates and initializes the array of ratings to 0
   b. The second constructor that takes the first name, the last name, the name of the book, the year of publication, and an array of 10 values and initializes everything accordingly.
3. (3 pts) A destructor that does what destructors do.
4. (3 pts) A method that calculates the average rating of the book by traversing the ratings array and finding the average rating. The method should return that rating (as a double).
5. (2 pts) A method that prints out the rating, and, if the rating is 0, prints "(no ratings yet)".
6. (2 pts) Write a method that prints out the book's information, including author information, year of publication, and average rating.
7. (3 pts) Write an operator overload for the > (or <) operator for your book class.

In your main:

1. (12 pts for working version) Create an array of 5 book objects (on the stack). Write a function that traverses the array of students and finds the "greatest" book object, using the overloaded operator method you wrote. Print out that book's information using the method for printing out book information
2. (3 pts for working version) Create one book object on the heap and print out that book's information using the method for printing out book's information
3. (3 pts for working version) Create an array of 2 book objects on the heap, and print out their information using the method for printing out book's information

Part B: Linked Lists:

For this it's probably easiest to create a new project for the linked list files. I'm giving you the .hpp files and the main, below. I'm also giving you the output my program gave me.

Your job is to fill in the code for the .cpp files for a linked list.

Given the following for SNode.hpp (with the data being a string):

SNode.hpp is defined as follows:

```
#ifndef SNODE_HPP_
#define SNODE_HPP_
#include <iostream>
using namespace std;
```

```
class SNode {
	friend class SLL;
	int rating;
	string comments;    // instead of int data, now the data is both the rating
	                    // and the accompanying comments – think of when you rate
	                    // a book or a song, and then are asked to share comments
	                    //about the book.  Both are the data in this node.
	SNode *next;
public:
	SNode();  // this constructor takes no input.  It and asks the user to input a
	          // rating, then reads the rating into the rating field, and
	          // then asks the user to input their comments, and reads in the
	          // comments into the comments field.
	SNode(int r, string c);
	          // this constructor takes as input an integer and a string, and
	          // initializes the rating field to the integer, and the comment
	          // field to the string.

	~SNode();
	void printNode();

};

#endif /* SNODE_HPP_ */
```

(4 pts) Write the accompanying SNode.cpp

SLL.hpp is defined as follows:

```
#ifndef SLL_HPP_
#define SLL_HPP_

#include "SNode.hpp"
#include <stdlib.h>
#include <iostream>
using namespace std;

class SLL {
	SNode *first;
	SNode *last;
	int size;
public:
	SLL();
	~SLL();
	void printSLL();
	// (4 pts for working) write a method that prints out the linked list
	void insertInOrder(int r, string c);
	//(6 pts for working)
	//This method creates an ordered linked list (ordered by the rating).  If the
	//linked list is empty, it calls addFirst with r and c.  If the rating is less
	//than the first node's rating in the list, it calls addAtFirst.  If the
	//rating is greater than the last node's rating, it calls the push method.
	//Otherwise it creates a new node using r and c, and inserts that node in
```

```
        //order into the linked list.
        void push(int r, string c);
        // (4 pts for working)
         // pushes a new node (with rating r and c comments) onto the end of the linked
        // list
         // (remember to reset the last pointer) – I called this from the
         // insertInOrder() method.
        void addAtFront(int r, string c);
        // (5 pts for working)
         // adds a new node (made from r and c) to the beginning of the
        // list (remember to reset the first pointer) – I called this from
        // insertInOrder
        void addFirst(int r, string c);
        //(3 pts for working)
        //adds the very first node (made from r and c) to an empty list
        // I called this from insertInOrder
        int pop();
        // (5 pts for working)
        // removes the last node from the linked list, and returns the rating from the
        // node removed.
        double getAve();
        // (4 pts for working)
        // calculates the average rating (by traversing the list) and then returns
        // the average rating.

};


#endif /* SLL_HPP_ */

/************MAIN FUNCTION ********************/

#include <iostream>
#include "SLL.hpp"
using namespace std;

int main() {
        int r[10] = {9,8,4,5,11,10,3,6,8,2};
        string s[10] = {"really good!","loved it","mediocre", "okay, not great","best
book ever!", "awesome!","boring","not bad","definitely worth reading", "terrible!"};
        SLL *list = new SLL();
        for (int i = 0; i < 10; i++) {
                list->insertInOrder(r[i],s[i]);
                list->printSLL();
        }
        cout << "Ave: "<< list->getAve() << endl;
        cout << "Popping " << list->pop() << endl;
        list->printSLL();
        cout << "Popping " << list->pop() << endl;
        list->printSLL();
        delete list;

        return 0;
}
```

```
/************************MY OUTPUT**************************/

(10 pts for getting same output)

Rating: 9,Comments: really good!


Rating: 8,Comments: loved it
Rating: 9,Comments: really good!


Rating: 4,Comments: mediocre
Rating: 8,Comments: loved it
Rating: 9,Comments: really good!


Rating: 4,Comments: mediocre
Rating: 5,Comments: okay, not great
Rating: 8,Comments: loved it
Rating: 9,Comments: really good!


Rating: 4,Comments: mediocre
Rating: 5,Comments: okay, not great
Rating: 8,Comments: loved it
Rating: 9,Comments: really good!
Rating: 11,Comments: best book ever!


Rating: 4,Comments: mediocre
Rating: 5,Comments: okay, not great
Rating: 8,Comments: loved it
Rating: 9,Comments: really good!
Rating: 10,Comments: awesome!
Rating: 11,Comments: best book ever!


Rating: 3,Comments: boring
Rating: 4,Comments: mediocre
Rating: 5,Comments: okay, not great
Rating: 8,Comments: loved it
Rating: 9,Comments: really good!
Rating: 10,Comments: awesome!
Rating: 11,Comments: best book ever!


Rating: 3,Comments: boring
Rating: 4,Comments: mediocre
Rating: 5,Comments: okay, not great
Rating: 6,Comments: not bad
Rating: 8,Comments: loved it
Rating: 9,Comments: really good!
Rating: 10,Comments: awesome!
Rating: 11,Comments: best book ever!
```

```
Rating: 3,Comments: boring
Rating: 4,Comments: mediocre
Rating: 5,Comments: okay, not great
Rating: 6,Comments: not bad
Rating: 8,Comments: definitely worth reading
Rating: 8,Comments: loved it
Rating: 9,Comments: really good!
Rating: 10,Comments: awesome!
Rating: 11,Comments: best book ever!


Rating: 2,Comments: terrible!
Rating: 3,Comments: boring
Rating: 4,Comments: mediocre
Rating: 5,Comments: okay, not great
Rating: 6,Comments: not bad
Rating: 8,Comments: definitely worth reading
Rating: 8,Comments: loved it
Rating: 9,Comments: really good!
Rating: 10,Comments: awesome!
Rating: 11,Comments: best book ever!


Ave: 6.6
removing 11, best book ever!
deleting 11, best book ever!
Popping 11
Rating: 2,Comments: terrible!
Rating: 3,Comments: boring
Rating: 4,Comments: mediocre
Rating: 5,Comments: okay, not great
Rating: 6,Comments: not bad
Rating: 8,Comments: definitely worth reading
Rating: 8,Comments: loved it
Rating: 9,Comments: really good!
Rating: 10,Comments: awesome!


removing 10, awesome!
deleting 10, awesome!
Popping 10
Rating: 2,Comments: terrible!
Rating: 3,Comments: boring
Rating: 4,Comments: mediocre
Rating: 5,Comments: okay, not great
Rating: 6,Comments: not bad
Rating: 8,Comments: definitely worth reading
Rating: 8,Comments: loved it
Rating: 9,Comments: really good!
```

```
deleting 2, terrible!
deleting 3, boring
deleting 4, mediocre
deleting 5, okay, not great
deleting 6, not bad
deleting 8, definitely worth reading
deleting 8, loved it
deleting 9, really good!
deleted each node in ll


/**********************************************************************/
```

Note that you've pretty much written the back end of a book app now. Doesn't it make more sense to store the ratings in a linked list than an array?

To turn in:

- The hpp, cpp, and main for your book class
- The hpp,cpp for the SNode class, the hpp,cpp for the SLL class