CS368 Programming Assignment 2

Section 1, Fall 2015

Due by 11:59 pm on Wednesday, November 4

In this page: Announcements | Overview | Specifications | Handing in Related pages: Assignments

Announcements

Check here periodically.

10/24/2015

- Program released. To ask questions about the homework and see questions posed by other students and their answers, go to: http://www.piazza.com/wisc/fall2015/cs3681 and sign-in using your wisc.edu account.
- Note: completing this assignment is critical to developing a good working understanding of pointers in C++. Start early and use good incremental code development with a driver program *prior to* using the main program that we've provided.

  You will be implementing a singly-linked list (also called a singly-linked chain of nodes) in this programming assignment. If you are not familiar with linked lists (or would like a refresher), be sure to check out this reading on linked lists and to contact Okan (by email, on Piazza, after class, or during office hours) with any questions you may have.

Overview

Why are we doing this program?

Description

For this assignment you will modify the database you developed in Programming Assignment 1 by keeping the the students in a linked list (i.e., chain of linked nodes) ordered by student ID. You'll also make your code more object-oriented by using C++ classes for the ordered linked list and to represent the student information stored in the list.

Goals

The goals of this assignment are to give you experience:

- manipulating pointers
- allocating and deallocating memory using `new` and `delete`
- defining and using C++ classes
- writing classes to fit a given specification

Specifications

What are the program requirements?

For this assignment you will complete the implementation files (i.e., the source files) for two classes for which we've provided the interface files (i.e., the header files). We've also provided an object file for the main program

that uses the classes that you implement and interacts with a user using commands similar to the ones in Programming Assignment 1.

The `Student` Class

The `Student` class is used to represent individual students. Each `Student` object will contain a student ID, a dynamic array of CourseInfo structures that encapsulate the course information of student and the number of courses student has taken. The header file containing the interface for the `Student` class is given in Student.h. You can copy this file from this location:

```
~cs368-1/public/html/assignments/p2/files/Student.h
```

Complete the `Student` class by writing all of the member functions as described in the header file.

The `SortedList` Class

The `SortedList` class is used to implement an ordered singly-linked list of students. The list is ordered by student ID from smallest to largest. The header file containing the interface for the `SortedList` class is given in SortedList.h. You can copy this file from this location:

```
~cs368-1/public/html/assignments/p2/files/SortedList.h
```

Complete the `SortedList` class by writing all of the member functions as described in the header file.

User Interface

We have provided you with the code for the main program, which you must make work with your code. You can copy the main program containing the user interface from this location:

```
~cs368-1/public/html/assignments/p2/files/studentDB.o
```

The user interface has already been compiled into object code. Since C++ object code is platform-dependent (unlike Java bytecode, which is platform-independent), the precompiled file of the user interface requires you to do your work on the `galapagos` Linux machines.

In order to create an executable for your program, you'll need to link the user interface object code with the object code for your classes. The following Linux commands show how this can is done:

```
g++ -c Student.cpp
g++ -c SortedList.cpp
g++ Student.o SortedList.o studentDB.o -o runDB
```

The first two commands compile the source files that you've implemented. The third command links the object files for this program and creates an executable named `runDB`.

The commands that our user interface accepts are similar to those for Programming Assignment 1 except that "`c`" command has a different function and there are additional commands, including the "help" command:

`a ID`	add a student to the database
`d ID`	delete a student from the database
`u ID CourseNumber Year Semester Credit Grade`	update a student's record to include a grade for a course that was offered in a specific year and semester
`r CourseNumber Year`	print the class roster of a course offered at a specific semester

`Semester`	
`c ID1 ID2 ID3 ...`	print the pairs of students in the given group who were classmates for at least one semester.
`p`	print the database
`q`	quit the program
`?`	print out help on the commands

Error and Bounds Checking

All error checking and bounds checking is done in the user interface object code given to you (except for the case of printing classmates, look at the header file for details). You do not need to add any. In particular, you may assume that all values passed to constructors or functions are valid and have the appropriate range as described in Programming Assignment 1.

Handing in

What should be handed in?

Make sure your code follows the style and commenting standards used in CS 302 and CS 367. Note: the commenting standards use javadoc comments for class, method, and constructor headers. You do not need to use javadoc comments in your programs for CS 368; however, your comments should include the same information as the javadoc comments. For example, your function header comments should include a description of what the function does, the name and a short description of each parameter, and a description of the return value.

Electronically submit the following files to your **`In`** "handin" directory by the due date and time (or refer to the late policy):

- **`Student.cpp`** containing your source code for the `Student` class, and
- **`SortedList.cpp`** containing your source code for the `SortedList` class.

- Pair Programming students must also submit a `README` file: All students working in pairs must read the collaboration policy and submit a `README` file. Each student working in a pair must hand-in this file to his/her own hand-in directory. Copy, complete, and hand-in the file that is linked here:
  - `README.txt`

Please do not turn in `Student.h`, `SortedList.h`, or `studentDB.o`. You should **not** modify these files in any way. We will use the original versions of these files to execute your code for grading.

Last Updated: 10/24/2015