Submit Your CODE Using CANVAS
11:59PM on Friday February 21

Submit Your DOCUMENTATION Using CANVAS
11:59PM on Friday February 7

< Important Boiler Plate >

Start early to give yourself an opportunity to ask questions!!!

Waiting until the last minute to begin the assignment is not recommended and will not get you an extension on the submission deadline.

Debugging issues are far easier to resolve in person when we can see exactly what you and your code are doing.

All CPE 212 projects are automatically graded in order to provide you timely feedback. So, it is critical that you follow all directions for the preparation and submission of your projects for grading. Failure to follow the directions may result in zero credit (0 points).

<Project03 Goals and Description>

The goal of this is to implement the baseline data storage for a gradebook. You will implement a **List** data structure to hold different types, and use it in a representation of a **Student** and a **Classroom** object.

As part of this assignment, you will once again practice implementing components that will be integrated to form a larger product. This time you will write a **List class** that will be integrated into a larger software product provided by the instructor. This **List class** uses is a **Forward List**, and uses a **Template** to use different types within this list. The list then allows you to **Append** or **Delete** a node based on the data. The data type implemented must have an **is_equals (==)** operator overload, as well as a **not_equal (!=)** overload. The list provides access to the list's internal data stored by the nodes through a series of iteration functions. **AtEnd** tells the user that the iterator has no more data to access, **IterateItem** returns the current value of the iterator and increments it forward, and **ResetIterator** to reset the iterator position to the start.

This list is used in two different objects, **Student** which holds grades of a student, including their Projects, Exams, and final exam grade. **This object has been fully implemented and provided by your instructor.** The second implementation of the list is the **Classroom**. The classroom will be implemented by the student, and simply holds a list of students. This object includes **AddStudent** which adds a student to the list as long as this student (checked by UID) is not already stored. **RemoveStudent** will then remove student (Checked by UID). Both functions will return false if they fail to perform their designated operation. Class then has 3 functions to **Add Projects, Exams,** and **FinalExam** grades to the students stored. These functions take a List of assignments as the input, will iterate through this list, and if the UID matches a student, that type of

grade is added to the Student. If the UID of the assignment is not found in the list, it is simply not added. It is not guaranteed that a student has completed / turned in an assignment (Much like this class) and it's not guaranteed that a student will be found that matches an assignment's UID.

- **Do not modify main.cpp, list.hpp, classroom.hpp, student.hpp, or student.cpp. There are also a few functions provided in classroom.cpp. DO NOT MODIFY THOSE EITHER**

You will write list_impl.hpp and classroom.cpp !!!

- All program output is performed by the code in the main.cpp file – do not include any output statements in the files you write!!!

The interfaces to each of these functions you must write are described in detail on subsequent pages and in the files provided

Hint: Match the output of your program to that of the sample solution!!!

Directions for running the sample solution appear on the following page.

Running the Sample Solution on blackhawk

The best description of what your code should do is the Sample Solution for the project.

Run the sample solution by typing the following at **blackhawk** terminal window command prompt

/home/facstaff/taf0004/CPE212SP20/Project03/Solution/P3 inputfilename

where **inputfilename** is the name of one of the provided input files (for example, ***P3input1.txt***). ***Your current working directory must contain the input files for this to work.***

Running the Preview Script on blackhawk

Run the preview script by typing the following in a **blackhawk** terminal window command prompt

/home/facstaff/taf0004/CPE212SP20/Project03/preview03.bash

This script will run both the Sample Solution AND your project03 executable program on the complete set of input files, and it compares the outputs of the two programs line by line to identify errors in your program's outputs. Make sure that the output of your program exactly matches the output of the Sample Solution.

<Project03 Include File Constraints>

You are not allowed to modify the include files within any file. All of the includes needed are already included.

Failure to follow these directions will result in zero (0) credit on this assignment.

<Project03 Object Description and Hints>

You should take the list implementation that we discussed in class. The following functions need to be implemented within the ***list_impl.hpp***:

```
void AppendItem(const Type &data);
bool DeleteItem(const Type &data);
unsigned int Count() const;

Type& Front();
Type Front() const;
Type& Back();
Type Back() const;

Type* IterateItems() const;
bool AtEnd() const;
void ResetIterator() const;
```

Since this is a template version of the list, you shouldn't worry about any specific implementation requirements other than verifying that the data searched for DeleteItem is equal.
For the iterators you should follow the same description we had in class about iterators, but remember you should return a pointer to the data in question, rather than a copy of the data. This is because we will need to use IterateItems() to modify the underlying data.
Another note is the mutable iterator node. Remember this is a key word that means it ignores the const-ness of the functions that use it.

Your other file to modify is classroom.cpp. You are given two functions already implemented:

```
void PrintClassroomInformation() const;
Student* find_student(unsigned int UID, bool &found);
```

find_student is an important function, which will search your list for the required student based on the UID of the student (think your student ID). This provides you a pointer to the underlying data of Student, rather than anything involving the node. **In order for**

this to work properly your list has to function correctly. If no student is found, the Boolean reference will return false, and true if a student is found.

You then have 5 functions in the Classroom object to implement

```
bool AddStudent(const std::string &firstName, const std::string &lastName,
                unsigned int UID);
bool RemoveStudent(unsigned int UID);


void AddProjects(const List<Assignment> &projects);
void AddExams(const List<Assignment> &exams);
void SetFinalExams(const List<Assignment> &finals);
```

AddStudent should construct a student (on the stack) and append it to the _classList. **IF THE STUDENT ALREADY EXISTS, THIS SHOULD FAIL AND RETURN FALSE.** RemoveStudent should remove a student based on the UID. Students are compared SOLELY on their UID, so use this for comparing if the student exists and for deleting. **For both of these functions put more work on the List object, and not on your specific class implementation.**

For the AddAssignment functions, you will get a list of assignments, these have a score and a UID attached to them. You should search for the student with a UID equal to the UID of the assignment. **It is not guaranteed that a student has completed / turned in an assignment (Much like this class) and its not guaranteed that a student will be found that matches an assignment's UID.**

A SPECIAL NOTE ABOUT CORRECTNESS: The test script will run your code using valgrind. Please read any notes about valgrind in the canvas -> modules -> tutorials -> valgrind. This will verify that you are cleaning up all memory correctly. **The valgrind check must pass in order for your code to be considered correct**. **FAILURE TO DO THIS WILL RESULT IN A 0 FOR THAT TEST**.

<Important Project03 Submission Instructions>

(1) **Follow all submission instructions provided on Canvas.**

(2) **Below is a list of the files you must submit. It is very important that you use the correct file names: [lowercase letters, no spaces, no underscores] in order for your program to compile and execute.**

List_impl.hpp(write and submit this file using the information from list.h)

Classroom.cpp (write and submit this file using the information from classroom.h)

A COMPLETE SUBMISSION is one that includes the specified file as an attachment to your submission. All other submissions will be considered to be incomplete.

INCOMPLETE SUBMISSIONS will receive zero credit (0 points) on this assignment.

DO NOT USE A FILE COMPRESSION PROGRAM (.zip, .tar, etc. are not allowed)

Failure to satisfy this requirement will result in zero credit (0 points) on this assignment.

(3) ***Do not modify or submit the file named main.cpp, classroom.hpp, student.cpp, student.hpp or list.hpp !!!***

Failure to satisfy this requirement will result in zero credit (0 points) on this assignment.

(4) ***All output to the monitor (stdout) will be performed by the code provided in main.cpp and the provided functions in your objects.***

NOTE: Your project submission will be automatically graded so it is EXTREMELY IMPORTANT that you READ and FOLLOW the project directions.

Submit only the specified files via the Project03 CANVAS dropbox.

Submissions by email receive ZERO CREDIT (0 points)!!!

Submissions that do not compile receive ZERO CREDIT (0 points)!!!