Project5

Due Mar 12, 2023 by 11:59pm

Design Due date is is before the implementation due date.

Project 5: Enrollment System

Goals: Working with multiple classes, different type of containers and sorting based on given criteria

Accept the assignment and create your own private GitHub repository with the started code by clicking on this link, (https://classroom.github.com/a/qAhryp-7). Do not change the repository name or make it public.

You can work on this assignment **individually or in pairs**. In all cases, you MUST either have a single name or two names at the top of the README file. If you are working as a pair, you should be working together on one Github Repository, so you should invite the second person as a collaborator to the repository. (Settings > Manage Access > Invite teams of people)

You will build an Enrollment System application which processes enrollments. The enrollment system can work with multiple universities. Only one university can be currently active at a time.

The main functions that need to be implemented are:

```
  // Add university to the enrollments system
  bool addUniversity(const string &name);

  // Set this university as the active university for other functions
  bool setCurrentUniversity(const string &name);

  // Return the current active university name
  string getUniversityName() const;

  // Read the student list for current active university
  // return true if file successfully read
  bool readStudentList(const string &filename);

  // Read the course list for current active university
  // return true if file successfully read
  bool readCourseList(const string &filename);

  // Read the student enrollment information for current active
university
```

```
  // return true if file successfully read
  bool readEnrollmentInfo(const string &filename);

  // Drop student from given course, return true if successful
  bool dropCourse(int studentID, const string &courseNumber);

  // Add student to the given course, return true if successful
  bool addCourse(int studentID, const string &courseNumber);

  // Return true if student is in the given course
  bool isInCourse(int studentID, const string &courseNumber) const;

  // Return the courses student is enrolled in
  // The returned courses are separated by commas and sorted by
course name
  string getEnrolledCourses(int studentID) const;

  // Return the title for the course
  string getCourseTitle(const string &courseNumber);

  // Return class list sorted by last name of students
  string getClassListByLastName(const string &courseNumber) const;

  // Return class list sorted by id of students
  string getClassListByID(const string &courseNumber) const;
```

See the tests in `main.cpp` script to better understand the return value for each function.

Classes and Data Structures:

Your EnrollmentSystem class must implement the public functions that have been provided. As part of your design, you can, but do not have to, add more classes. I suggest the following classes:

Student - store student information, implement operator<<, dropCourse, addCourse, isInCourse functions and have a container of Course pointers that the student is currently enrolled in.

Course - store course information, a container of Student pointers, and functions to addStudent and removeStudent from the course. This function can also implement getClassListByLastName and getClassListByID. Since the last two function require the data to be sorted in different ways, static comparator functions, such as cmpLastName and cmpID, that you will write can be helpful.

University - store name of university, container of Student pointers, container of Course pointers. University class will be responsible for deleting all the pointers in the destructor. This class also can implement readStudentList, readCourseList and readEnrollmentInfo to be called from EnrollmentSystem class. It is best if EnrollmentSystem does not have direct access to Student and Course classes to maintain modularity. University class can implement addCourse, dropCourse, isInCourse, getEnrolledCourses, getClassListByLastName, getClassListByID and getCourseTitle so EnrollmentSystem can call them.

EnrollmentSystem - store a container of pointers to known universities and the current active university. Uses the stores university object to make calls and make changes to the enrollment.

Tips:

- If a function does not need to modify the object, declare it as const
- The map container is very useful when you have a key (such as idNumber for Student or courseNumber for Course objects).
- You can erase an object from map ,(https://cplusplus.com/reference/map/map/erase/) using the key.
- When reading text files, a common pattern is `while (inputStream >> someVariable) { do things }`
- You can sort a vector based on different criteria using sort(result.begin(), result.end(), cmpLastName); The function `cmpLastname` will need to be static, take 2 objects and return true/false.

Submit:

1. Run `./create-output.sh > output.txt 2>&1` and examine the `output.txt` file for any issues that need fixing
2. Submit to Canvas a zip file of your code as downloaded from the GitHub repository

Grading Rubric:

Good Design: 5/25

Completeness and correctness 15/25

Coding standards 5/25

Deductions for unnecessary globals, poor design choices, incorrectly named files, etc.