CSE 108 (January 2025)

Offline 02

Deadline: Sunday, May 11 (11:55 PM)

The goal of this assignment is to implement a basic academic management system using object-oriented programming in C++. The assignment simulates a real-world scenario where each student can register for multiple courses, track their grades, calculate CGPA based on credit hours, and perform various queries such as identifying favorite courses and academic toppers. In this system, the grade points are calculated on a scale between 0 and 4.0, and **a student can pass a course only if he/she can obtain a grade point greater than or equal to 2.0 in that course**. *For storing strings, you can use the C++ <string> library if convenient.*

Tasks:

1. Create a class named **`Course`** with the following private member variables**:**
   - `string name` - name of the course
   - `float creditHour` - number of credit hours assigned to this course
2. Implement the following constructors for the **`Course`** class:
   - Default constructor
   - Parameterized constructor: `Course(string name, float creditHour)`
3. Implement the following public member functions in the **`Course`** class:
   - `string getName()` - returns the name of the course
   - `float getCreditHour()` - returns the number of credit hours assigned
   - `void setName(string name)` - sets the name of the course
   - `void setCreditHour(float creditHour)` - sets the credit hours of the course
   - `void display()` - prints the name and credit hours of the course
4. Create a class named **`Student`** with the following private member variables**:**
   - `string name` - name of the student
   - `int id` - ID of the student
   - `Course* courses` - a dynamically allocated array of courses registered by the student
   - `int totalCourses` - total number of courses registered by the student
   - `int maxCourses` - maximum number of courses a student can register
   - `float* gradePoints` - a dynamically allocated array containing marks for the respective courses
5. Implement the following constructors and destructor for the **`Student`** class:
   - Default constructor
   - Parameterized constructor: `Student(string name, int id, int maxCourses)`

    - A copy constructor
    - A destructor
6. Implement the following public member functions in the **`Student`** class:
    - `void setName(string name)` - sets the name of the student
    - `void setId(int id)` - sets the ID of the student
    - `void setInfo(string name, int id)` - sets the name and ID of the student
    - `void addCourse(Course c)` - registers course c for the student, and by default sets the gradePoint obtained for this course to 0
    - `void addCourse(Course course, float gradePoint)` - registers course c for the student, and sets the corresponding gradePoint obtained for this course
    - `void setGradePoint(Course c, float gradePoint)` - sets the gradePoint obtained by the student in a particular course c
    - `void setGradePoint(float* gradePoints, int n)` - sets the gradePoints for the first `n` courses registered by the student
    - `string getName()` - returns the name of the student
    - `float getCGPA()` - computes and returns the CGPA of the student (calculated considering respective credit hours and assigning weights accordingly)
    - `float getGradePoint(Course c)` - returns the grade point obtained by the student in a particular course c
    - `int getTotalCourses()` - returns the total number of courses registered by the student
    - `float getTotalCreditHours()` - returns the total credit hours earned by the student (a student earns the corresponding credit hours of a course only after passing it successfully)
    - `Course getMostFavoriteCourse()` - returns the course where the student achieved the highest grade point among all of his/her courses; tie is broken by the order of course registration
    - `Course getLeastFavoriteCourse()` - returns the course where the student achieved the lowest gradePoint among all of his/her courses; tie is broken by the order of course registration
    - `Course* getFailedCourses(int &count)` - returns a list of courses in which the student has failed
    - `void display()` - prints the student's name, ID, course-wise obtained grade points, CGPA, total credits earned, the most favorite and the least favorite course
7. Declare a global array named `students` to hold pointers to `Student` objects, and an integer variable `totalStudents` to track the current number of students in the system..
8. Implement the following global functions:
    - `Student getTopper()` - returns the student who obtained the highest CGPA among all the students in the `students` array

- `Student getTopper(Course c)` - returns the student who obtained the highest grade point in the course c among all the students in the `students` array who enrolled in course `c`

9. Please carefully see the sample main function to understand the output format. You can copy the main function directly to your code. **You must be able to produce your output in the same format as the Expected Output for the sample main function.**
10. You **must** check against erroneous arguments wherever necessary and print suitable error messages.
11. You are NOT allowed to define any public member functions other than the ones mentioned above for either class, not even getters and setters other than the ones mentioned.
12. You are NOT allowed to use any friend class or friend function in your code.

Sample `main()` Function:

```
int main() {
    // generate courses
    const int COURSE_COUNT = 6;
    Course courses[COURSE_COUNT] = {
        Course("CSE107", 3),
        Course("CSE105", 3),
        Course("CSE108", 1.5),
        Course("CSE106", 1.5),
        Course("EEE164", 0.75),
        Course("ME174", 0.75),
    };
    float gradePoints[COURSE_COUNT] = {4.0, 4.0, 3.5, 3.5, 4.0, 3.25};

    // generate students
    Student s1 = Student("Sheldon", 1, 5);
    students[totalStudents++] = &s1;

    // add courses to s1
    s1.addCourse(courses[0]);
    s1.addCourse(courses[1]);
    s1.addCourse(courses[2]);
    s1.addCourse(courses[3]);
    s1.addCourse(courses[4]);
    s1.addCourse(courses[5]);
    s1.setGradePoint(gradePoints, s1.getTotalCourses());
    s1.display();

    Student s2 = Student("Penny", 2, 5);
    students[totalStudents++] = &s2;
    s2.addCourse(courses[0]);
    s2.addCourse(courses[2]);
    s2.addCourse(courses[5]);
    s2.setGradePoint(gradePoints, s2.getTotalCourses());
    s2.setGradePoint(courses[0], 3.25);
    s2.display();

    Student s3 = s2;
    students[totalStudents++] = &s3;
    s3.setName("Leonard");
    s3.setId(3);
    s3.setGradePoint(gradePoints, s3.getTotalCourses());
    s3.addCourse(courses[1], 3.75);
    s3.display();
```

```
    Student s4 = s3;
    students[totalStudents++] = &s4;
    s4.setInfo("Howard", 4);
    s4.setGradePoint(gradePoints, s4.getTotalCourses());
    s4.addCourse(courses[3], 3.75);
    s4.display();

    Student s5 = s4;
    students[totalStudents++] = &s5;
    s5.setInfo("Raj", 5);
    s5.setGradePoint(gradePoints, s5.getTotalCourses());
    s5.setGradePoint(courses[0], 1.5);
    s5.setGradePoint(courses[2], 2.0);
    s5.setGradePoint(courses[5], 1.75);
    s5.setGradePoint(courses[3], 3.75);
    s5.display();

    int failedCount;
    Course* failedCourses = s5.getFailedCourses(failedCount);
    cout << "Failed Courses for " << s5.getName() << ":" << endl;
    for (int i = 0; i < failedCount; ++i) {
        failedCourses[i].display();
        cout << endl;
    }
    delete[] failedCourses;

    cout << "==================================" << endl;
    Student topper = getTopper();
    cout << "Topper: " << topper.getName() << endl;
    cout << "Topper CGPA: " << topper.getCGPA() << endl;
    cout << "==================================" << endl;

    for (int i = 0; i < COURSE_COUNT; ++i) {
        Course c = courses[i];
        Student topperInCourse = getTopper(c);
        cout << "Topper in " << c.getName() << ": " <<
topperInCourse.getName() << endl;
        cout << "Topper in " << c.getName() << " gradePoint: " <<
topperInCourse.getGradePoint(c) << endl;
        cout << "==================================" << endl;
    }

    return 0;
}
```

Expected Output:

```
Cannot add more courses to Sheldon
==================================
Student Name: Sheldon, ID: 1
Course Name: CSE107, Credit Hour: 3, gradePoint: 4
Course Name: CSE105, Credit Hour: 3, gradePoint: 4
Course Name: CSE108, Credit Hour: 1.5, gradePoint: 3.5
Course Name: CSE106, Credit Hour: 1.5, gradePoint: 3.5
Course Name: EEE164, Credit Hour: 0.75, gradePoint: 4
CGPA: 3.84615
Total Credit Hours Earned: 9.75
Most Favorite Course: CSE107
Least Favorite Course: CSE108
==================================
==================================
Student Name: Penny, ID: 2
Course Name: CSE107, Credit Hour: 3, gradePoint: 3.25
Course Name: CSE108, Credit Hour: 1.5, gradePoint: 4
Course Name: ME174, Credit Hour: 0.75, gradePoint: 3.5
CGPA: 3.5
Total Credit Hours Earned: 5.25
Most Favorite Course: CSE108
Least Favorite Course: CSE107
==================================
==================================
Student Name: Leonard, ID: 3
Course Name: CSE107, Credit Hour: 3, gradePoint: 4
Course Name: CSE108, Credit Hour: 1.5, gradePoint: 4
Course Name: ME174, Credit Hour: 0.75, gradePoint: 3.5
Course Name: CSE105, Credit Hour: 3, gradePoint: 3.75
CGPA: 3.86364
Total Credit Hours Earned: 8.25
Most Favorite Course: CSE107
Least Favorite Course: ME174
==================================
==================================
Student Name: Howard, ID: 4
Course Name: CSE107, Credit Hour: 3, gradePoint: 4
Course Name: CSE108, Credit Hour: 1.5, gradePoint: 4
Course Name: ME174, Credit Hour: 0.75, gradePoint: 3.5
Course Name: CSE105, Credit Hour: 3, gradePoint: 3.5
Course Name: CSE106, Credit Hour: 1.5, gradePoint: 3.75
CGPA: 3.76923
Total Credit Hours Earned: 9.75
```

```
Most Favorite Course: CSE107
Least Favorite Course: ME174
==================================
==================================
Student Name: Raj, ID: 5
Course Name: CSE107, Credit Hour: 3, gradePoint: 1.5
Course Name: CSE108, Credit Hour: 1.5, gradePoint: 2
Course Name: ME174, Credit Hour: 0.75, gradePoint: 1.75
Course Name: CSE105, Credit Hour: 3, gradePoint: 3.5
Course Name: CSE106, Credit Hour: 1.5, gradePoint: 3.75
CGPA: 2.55769
Total Credit Hours Earned: 6
Most Favorite Course: CSE106
Least Favorite Course: CSE107
==================================
Failed Courses for Raj:
Course Name: CSE107, Credit Hour: 3
Course Name: ME174, Credit Hour: 0.75
==================================
Topper: Leonard
Topper CGPA: 3.86364
==================================
Topper in CSE107: Sheldon
Topper in CSE107 gradePoint: 4
==================================
Topper in CSE105: Sheldon
Topper in CSE105 gradePoint: 4
==================================
Topper in CSE108: Penny
Topper in CSE108 gradePoint: 4
==================================
Topper in CSE106: Howard
Topper in CSE106 gradePoint: 3.75
==================================
Topper in EEE164: Sheldon
Topper in EEE164 gradePoint: 4
==================================
Topper in ME174: Penny
Topper in ME174 gradePoint: 3.5
==================================
```

Mark Distribution

Correct Implementation of the Course Class	10%
Correct implementation of the Student Class Copy Constructor	10%
Correct Implementation of the rest of the Student Class	50%
Proper Allocation and Deallocation of Memory	10%
Correct Implementation of the Global Functions	10%
Correct Formatting of the Output	5%
Proper Handling of Erroneous Arguments to Functions and Printing Suitable Error Messages	5%

Submission guidelines:

1. Create a folder named after your student ID.

2. Write your code in a single file named ***your_ID.cpp***.

3. Move the .cpp file into the folder from step1.

4. Zip the folder and name it after your student ID. Submit this zip on Moodle.

For example,

2305xyz/

└──2305xyz.cpp

Then submit 2305xyz.zip

Deadline: Sunday, May 11 2025 (11:55PM)

Please do not copy from any sources (friends, internet, AI tools like ChatGPT, etc.). Doing so will result in severe penalties.