CS 201, Spring 2023
Homework Assignment 3

Due: 23:59, May 25, 2023

In this assignment, you will implement a simple registration system by using linked lists. The registration system stores information about students and courses. For each student, the system stores an id, first name, last name, and a list of her/his course enrollments. Each course is represented by its id and title. This system **MUST** use a sorted linear doubly linked list with no dummy head node to store the students, and for each student, a sorted linear singly linked list with no dummy head node to store the course enrollments for that student. The students are stored in ascending order of their ids. The courses are stored in ascending order of their ids.

The registration system will have the following functionalities; the details of these functionalities are given below:

1. Add a student
2. Delete a student
3. Add a course for a student
4. Withdraw a student from a course
5. Cancel a course
6. Show detailed information about a particular student
7. Show detailed information about a particular course
8. Show all students

Add a student: The registration system will allow to add a new student indicating her/his student id, first name, and last name. Since the student ids are unique positive integers, the system should check whether or not the specified student id already exists (i.e., whether or not it is the id of an existing student), and if the student id exists, it should not allow the operation and display a warning message. The list must remain sorted by student id after this operation.

Delete a student: The registration system will allow to delete an existing student indicating her/his student id. If the student does not exist (i.e., if there is no student with the specified id), the system should display a warning message. Note that this operation will also drop all courses in which the student was enrolled.

Add a course for a student: The registration system will allow to add a new course for a particular student. For that, the student id, the course id, and the course name have to be specified. The system should check whether or not this student exists; if she/he does not, it should prevent to add a course and display a warning message. If the student exists and the student is not already enrolled in this course, the given course is added to student's course list. The system should check whether or not this course id exists; if it exists, it should prevent to add this course and display a warning message. The courses are also stored sorted.

Withdraw a student from a course: The registration system will allow to delete an existing course indicating its course id from a student's course enrollment list. If the student does not exist (i.e., if there is no student with the specified id) or the student is not enrolled in this course (i.e., if there is no course with the specified id), the system should display a warning message.

Cancel a course: The registration system will allow to delete an existing course indicating its course id. Note that this operation will remove the course from the course enrollment lists for all students. If the course does not exist (i.e., if there is no course with the specified id), the system should display a warning message.

Show detailed information about a particular student: The registration system will allow to specify a student id and display detailed information about that particular student. This information includes the student id, the student name, the list of courses enrolled by this student including the course id and the course name for each course. The courses will be displayed in ascending order of their ids. If the student does not exist (i.e., if there is no student with the specified student id), the system should display a warning message.

Show detailed information about a particular course: The registration system will allow to specify a course id and display detailed information about that particular course. This information includes the course id, the course name, the list of students enrolled in this course including the student id and the student name for each student. The students will be displayed in ascending order of their ids. If the course does not exist (i.e., if there is no course with the specified course id), the system should display a warning message.

Show the list of all students: The registration system will allow to display a list of all the students. This list includes the student id, the student name, and the list of courses enrolled by each student. The students will be displayed in ascending order of their ids. Also, the students will be displayed in ascending order of their ids for each student.

Below is the required `public` part of the `RegistrationSystem` class that you must write in this assignment. The name of the class must be `RegistrationSystem`, and must include these public member functions. We will use these functions to test your code. The interface for the class must be written in a file called `RegistrationSystem.h` and its implementation must be written in a file called `RegistrationSystem.cpp`. You can define additional public and private member functions and data members in this class. You can also define additional classes in your solution.

```
class RegistrationSystem {
public:
    RegistrationSystem();
    ~RegistrationSystem();

    void addStudent( const int studentId, const string firstName, const string
        lastName );
    void deleteStudent( const int studentId );

    void addCourse( const int studentId, const int courseId, const string
        courseName );
    void withdrawCourse( const int studentId, const int courseId );
    void cancelCourse( const int courseId );

    void showStudent( const int studentId );
    void showCourse( const int courseId );
    void showAllStudents();
};
```

Here is an example test program that uses this class and the corresponding output. We will use a similar program to test your solution so make sure that the name of the class is `RegistrationSystem`,

its interface is in the file called `RegistrationSystem.h`, and the required functions are defined as shown above.

Example test code:

```
#include "RegistrationSystem.h"

int main() {

    RegistrationSystem rs;

    rs.showAllStudents();
    cout << endl;

    rs.addStudent(2000, "Kemal", "Ak");
    rs.addStudent(1000, "Nuri", "Yazici");
    rs.addStudent(4000, "Cengiz", "Erdem");
    rs.addStudent(3000, "Osman", "Top");
    rs.addStudent(4000, "Can", "Gezici");
    rs.addStudent(6000, "Can", "Gezici");
    rs.addStudent(5000, "Ali", "Akdere");
    rs.addStudent(7000, "Burcin", "Temiz");
    cout << endl;

    rs.showAllStudents();
    cout << endl;

    rs.addCourse(2000, 555, "CS555");
    rs.addCourse(2000, 540, "CS540");
    rs.addCourse(2000, 513, "CS513");
    rs.addCourse(2000, 524, "CS524");

    rs.addCourse(3000, 524, "CS524");
    rs.addCourse(3000, 540, "CS540");

    rs.addCourse(1000, 540, "CS540");
    rs.addCourse(1000, 524, "CS524");

    rs.addCourse(4000, 524, "CS524");
    rs.addCourse(4000, 510, "CS510");
    rs.addCourse(4000, 540, "CS540");
    rs.addCourse(4000, 513, "CS513");

    rs.addCourse(5000, 510, "CS510");
    rs.addCourse(5000, 513, "CS513");
    rs.addCourse(5000, 540, "CS540");

    rs.addCourse(6000, 540, "CS540");

    rs.addCourse(7000, 510, "CS510");
    rs.addCourse(7000, 513, "CS513");
```

```
    rs.addCourse(7000, 540, "CS540");

    rs.addCourse(3000, 524, "CS524");
    cout << endl;

    rs.deleteStudent(5000);
    rs.deleteStudent(5000);
    cout << endl;

    rs.showStudent(1000);
    rs.showStudent(3000);
    rs.showStudent(5000);
    cout << endl;

    rs.showAllStudents();
    cout << endl;

    rs.withdrawCourse(3000, 524);
    rs.withdrawCourse(2000, 555);
    rs.withdrawCourse(2000, 550);
    rs.withdrawCourse(10000, 510);
    cout << endl;

    rs.cancelCourse(540);
    rs.cancelCourse(201);
    cout << endl;

    rs.showCourse(524);
    rs.showCourse(540);
    rs.showStudent(7000);
    cout << endl;

    rs.deleteStudent(7000);
    cout << endl;

    rs.showStudent(3000);
    cout << endl;

    rs.showAllStudents();
    cout << endl;

    return 0;
}
```

Output of the example test code:

```
There are no students in the system

Student 2000 has been added
Student 1000 has been added
Student 4000 has been added
```

```
Student 3000 has been added
Student 4000 already exists
Student 6000 has been added
Student 5000 has been added
Student 7000 has been added

Student id First name Last name
1000 Nuri Yazici
2000 Kemal Ak
3000 Osman Top
4000 Cengiz Erdem
5000 Ali Akdere
6000 Can Gezici
7000 Burcin Temiz

Course 555 has been added to student 2000
Course 540 has been added to student 2000
Course 513 has been added to student 2000
Course 524 has been added to student 2000
Course 524 has been added to student 3000
Course 524 already exists with another name
Course 540 has been added to student 3000
Course 540 has been added to student 1000
Course 524 has been added to student 1000
Course 524 has been added to student 4000
Course 510 has been added to student 4000
Course 540 has been added to student 4000
Course 513 has been added to student 4000
Course 510 has been added to student 5000
Course 513 has been added to student 5000
Course 540 has been added to student 5000
Course 540 has been added to student 6000
Course 510 has been added to student 7000
Course 513 has been added to student 7000
Course 540 has been added to student 7000
Student 3000 is already enrolled in course 524

Student 5000 has been deleted
Student 5000 does not exist

Student id First name Last name
1000 Nuri Yazici
Course id Course name
524 CS524
540 CS540
Student id First name Last name
3000 Osman Top
Course id Course name
524 CS524
540 CS540
Student 5000 does not exist

Student id First name Last name
1000 Nuri Yazici
Course id Course name
524 CS524
```

```
540 CS540
2000 Kemal Ak
Course id Course name
513 CS513
524 CS524
540 CS540
555 CS555
3000 Osman Top
Course id Course name
524 CS524
540 CS540
4000 Cengiz Erdem
Course id Course name
510 CS510
513 CS513
524 CS524
540 CS540
6000 Can Gezici
Course id Course name
540 CS540
7000 Burcin Temiz
Course id Course name
510 CS510
513 CS513
540 CS540

Student 3000 has been withdrawn from course 524
Student 2000 has been withdrawn from course 555
Student 2000 is not enrolled in course 550
Student 10000 does not exist

Course 540 has been cancelled
Course 201 does not exist

Course id Course name
524 CS524
Student id First name Last name
1000 Nuri Yazici
2000 Kemal Ak
4000 Cengiz Erdem

Course 540 does not exist

Student id First name Last name
7000 Burcin Temiz
Course id Course name
510 CS510
513 CS513

Student 7000 has been deleted

Student id First name Last name
3000 Osman Top

Student id First name Last name
1000 Nuri Yazici
```

```
Course id Course name
524 CS524
2000 Kemal Ak
Course id Course name
513 CS513
524 CS524
3000 Osman Top
4000 Cengiz Erdem
Course id Course name
524 CS524
510 CS510
513 CS513
6000 Can Gezici
```

IMPORTANT NOTES:
Do not start your homework before reading these notes!!!

NOTES ABOUT IMPLEMENTATION:

1. You ARE NOT ALLOWED to modify the given parts of the header file. You MUST use linked lists in your implementation. You will get no points if you use dynamic or fixed-sized arrays or any other data structures such as vectors/arrays from the standard library. However, if necessary, you may define additional data members and member functions. You may also define additional classes.
2. Moreover, you ARE NOT ALLOWED to use any global variables or any global functions.
3. Output message for each operation MUST match the format shown in the output of the example code.
4. The blank lines in the expected output of the test code given above is shown for readability of the output and will be ignored during auto-grading.
5. You should NOT use **printf** function to display the output messages because it causes problems during auto-grading. Instead, display the messages using **cout**.
6. Your code MUST use linked lists to store student and course data in your implementations. You must use your own implementations of linked lists. You can adapt the codes and pseudocodes in the Carrano book. However, you are NOT allowed to use any existing codes from other sources (including the codes given in other textbooks, found on the internet, and belonging to your classmates, etc.). Furthermore, you are NOT allowed to use any data structure or related function from the C++ standard template library (STL).
7. Your code MUST NOT have any memory leaks. You will lose points if you have memory leaks in your program even though the outputs of the operations are correct. To detect memory leaks, you may want to use Valgrind which is available at `http://valgrind.org`.
8. Otherwise stated in the description, you may assume that the inputs for the functions are always valid (e.g., student id is a valid positive integer number) so that you do not need to make any input checks.

NOTES ABOUT SUBMISSION:

1. In this assignment, you must have separate interface and implementation files (i.e., separate `.h` and `.cpp` files) for your class. We will test your implementation by writing our own driver `.cpp` file which will include your header file. For this reason, your class' name MUST BE "`RegistrationSystem`" and your files' name MUST BE "`RegistrationSystem.h`" and "`RegistrationSystem.cpp`". You should submit these two files (and any additional files if you wrote additional classes in your solution) **as a single archive file (.zip file)**.

2. The code (`main` function) given above is just an example. We will test your implementation using different scenarios, which will contain different function calls. Thus, do not test your implementation only by using this example code. We recommend you to write your own driver files to make extra tests. However, you MUST NOT submit these test codes (we will use our own test code). In other words, do not submit a file that contains a function named `main`.

3. You should put all of your `.h` and `.cpp` files into a folder and zip the folder (in this zip file, there should not be any file containing a `main` function). The name of this zip file should conform to the following name convention: secX-Firstname-Lastname-StudentID.zip where X is your section number. The submissions that do not obey these rules will not be graded.

4. Make sure that each file that you submit (each and every file in the archive) contains your name, section, and student number at the top as comments.

5. You are free to write your programs in any environment (you may use Linux, Windows, MacOS, etc.). On the other hand, we will test your programs on "dijkstra.ug.bcc.bilkent.edu.tr" and we will expect your programs to compile and run on the dijkstra machine. If we could not get your program properly work on the dijkstra machine, you would lose a considerable amount of points. Thus, we recommend you to make sure that your program compiles and properly works on dijkstra.ug.bcc.bilkent.edu.tr before submitting your assignment.

6. This assignment is due by 23:59 on Tuesday, May 25, 2023. You should upload your work to Moodle before the deadline. No hardcopy submission is needed. The standard rules about late homework submissions apply. Please see the course home page for further discussion of the late homework policy.

7. We use an automated tool as well as manual inspection to check your submissions against plagiarism. Please see the course home page for further discussion of academic integrity and the honor code for programming courses in our department.

8. This homework will be graded by your TA Hakan Sivük (hakan.sivuk at bilkent.edu.tr). Thus, you may ask your homework related questions directly to him. There will also be a forum on Moodle for questions.