

CS 201, Fall 2010

Homework Assignment 4

Due: 18:00, December 23, 2010

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 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 an unsorted list.

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 courses are stored unsorted.

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. 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. 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.

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 );

    int 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, "Esra", "Akbas");
rs.addStudent(1000, "Mehmet", "Celik");
rs.addStudent(4000, "Gokhan", "Akca");
rs.addStudent(3000, "Fatih", "Isler");
rs.addStudent(4000, "Can", "Koyuncu");
rs.addStudent(6000, "Can", "Koyuncu");
rs.addStudent(5000, "Ali", "Akdere");
rs.addStudent(7000, "Burak", "Tosun");
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	Mehmet	Celik
2000	Esra	Akbaz
3000	Fatih	Isler
4000	Gokhan	Akcay
5000	Ali	Akdere
6000	Can	Koyuncu
7000	Burak	Tosun

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 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	Mehmet	Celik
	Course id	Course name
	540	CS540
	524	CS524

Student id	First name	Last name
3000	Fatih	Isler
	Course id	Course name
	524	CS524
	540	CS540

Student 5000 does not exist

Student id	First name	Last name
1000	Mehmet	Celik
	Course id	Course name
	540	CS540
	524	CS524
2000	Esra	Akbaz
	Course id	Course name
	555	CS555
	540	CS540
	513	CS513
	524	CS524
3000	Fatih	Isler
	Course id	Course name
	524	CS524
	540	CS540
4000	Gokhan	Akcay
	Course id	Course name
	524	CS524
	510	CS510
	540	CS540
	513	CS513
6000	Can	Koyuncu
	Course id	Course name
	540	CS540
7000	Burak	Tosun
	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	Mehmet	Celik
2000	Esra	Akbas
4000	Gokhan	Akcay

Course 540 does not exist

Student id	First name	Last name
7000	Burak	Tosun

Course id	Course name
510	CS510
513	CS513

Student 7000 has been deleted

Student id	First name	Last name
3000	Fatih	Isler

Student id	First name	Last name
1000	Mehmet	Celik

Course id	Course name
524	CS524

Student id	First name	Last name
2000	Esra	Akbas

Course id	Course name
513	CS513
524	CS524

Student id	First name	Last name
3000	Fatih	Isler

Student id	First name	Last name
4000	Gokhan	Akcay

Course id	Course name
524	CS524
510	CS510
513	CS513

Student id	First name	Last name
6000	Can	Koyuncu

Notes:

1. This assignment is due by 18:00 on Thursday, December 23th. You should upload your homework using the online submission form on the course web page before the deadline. No hardcopy submission is needed. The standard rules about late homework submissions apply. Please see the course syllabus for further discussion of the late homework policy as well as academic integrity. For those who did not return the signed copy of the “Honor Code Statement”, this homework will not be graded.
2. For this assignment, you must use your own implementation of linked lists. In other words, you cannot use any existing linked list code from other sources such as the `list` class in the C++ standard template library (STL). However, you can adapt the linked list codes in the Carrano book. You will get no points if you do not use linked lists as indicated.

3. 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.
4. 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 upload these two files (and any additional files if you wrote additional classes in your solution) as a single archive file (e.g., zip, tar, rar). The submissions that do not obey these rules will not be graded. We also recommend you to write your own driver file to test each of your functions. However, you MUST NOT submit this test code (we will use our own test code). In other words, your submitted code should not include any `main` function.
5. You are free to write your programs in any environment (you may use either Linux or Windows). 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. Therefore, we recommend you to make sure that your program compiles and properly works on "dijkstra.ug.bcc.bilkent.edu.tr" before submitting your assignment.