CS 201, Spring 2020
Homework Assignment 1

Due: 23:55, April 3, 2020

In this assignment, you will implement a simple registration system by using dynamic memory management. 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 dynamically allocated array to store the students, and for each student, another dynamically allocated array 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 array. The arrays must be allocated dynamically and must use memory only as much as needed (in other words, you cannot assume a maximum size for students and courses to allocate arrays that are larger than needed).

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

- Add a student
- Delete a student
- Add a course for a student
- Withdraw a student from a course
- Cancel a course
- Show detailed information about a particular student
- Show detailed information about a particular course
- Show all students

This assignment has two parts, whose requirements will be explained below.

PART A: **To take the final exam, you MUST submit at least this part and MUST get at least half of its points.**
This part includes the following four functionalities:

- Add a student
- Delete a student
- Show detailed information about a particular student
- Show all students

The details of these four functionalities will be given below. Note that this part does not involve any course information for the students. Therefore, in this part, the "Show detailed information about a particular student"or the "Show all students" functionalities will not be expected to display the list of the courses enrolled by the student(s).

PART B: In this part, you will complete the implementation of the entire registration system (all of the 8 functionalities listed above).

The details of all of the functionalities are given below:

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. Moreover, if the first name or the last name are empty, it should not allow the operation and display a warning message. The array 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 array. If the student is already enrolled in this course, it should display an appropriate message and the given course should not be added again. 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 array. 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 arrays 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 );

    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. The example test code provided below is just a sample. You must test your program with different inputs as well.

Example test code:

```
#include "RegistrationSystem.h"

int main() {

    RegistrationSystem rs;

    rs.showAllStudents();
    cout << endl;

    rs.addStudent( 2000, "Esra", "Kara" );
    rs.addStudent( 1000, "Mehmet", "Celik" );
    rs.addStudent( 4000, "Ahmet", "Akcay" );
    rs.addStudent( 3000, "Fatih", "Isler" );
    rs.addStudent( 4000, "Can", "Uzun" );
    rs.addStudent( 6000, "Can", "Uzun" );
    rs.addStudent( 5000, "Ali", "Akdere" );
    rs.addStudent( 7000, "Berrak", "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        Kara
3000        Fatih       Isler
4000        Ahmet      Akcay
5000        Ali         Akdere
6000        Can         Uzun
7000        Berrak       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        Kara
        Course id   Course name
        555         CS555
        540         CS540
        513         CS513
        524         CS524
3000        Fatih       Isler
        Course id   Course name
        524         CS524
        540         CS540
```

```
4000        Ahmet      Akcay
        Course id   Course name
        524         CS524
        510         CS510
        540         CS540
        513         CS513
6000        Can         Uzun
        Course id   Course name
        540         CS540
7000        Berrak       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        Kara
        4000        Ahmet      Akcay
Course 540 does not exist
Student id  First name  Last name
7000        Berrak       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
2000        Esra        Kara
        Course id   Course name
        513         CS513
        524         CS524
3000        Fatih       Isler
4000        Ahmet      Akcay
        Course id   Course name
        524         CS524
        510         CS510
        513         CS513
6000        Can         Uzun
```

Notes:

1. This assignment is due by 23:55 on Friday, April 3, 2020. You should upload your homework using the upload link on Moodle before the deadline. This upload page will be available between Mar 15 and April 6. 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.

2. Your code must use dynamically allocated arrays using pointers to store student and course data. You will get no points if you use static arrays with fixed sizes or other data structures such as `vector` from the standard library.

3. You cannot use any global variables in your implementation.

4. 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.

5. 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.

6. The *sample-testcode.cpp* driver file is given as sample test program. It does not test all aspects of your implementation. Thus, we will use other driver files (other test cases) to grade your assignment. Thus, you have to test your implementation thoroughly by writing your own driver files. Indeed, knowing how to test a program is crucial in designing and implementing any software. Thus, as a part of this homework, every student needs to experience (and hopefully to learn) how to test a simple program.

7. You are free to write your programs in any environment (you may use either Linux, Mac 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.

8. This homework will be graded by your TA **Alper Kağan Kayalı** (kagan.kayali AT bilkent.edu.tr). Thus, you may ask your homework related questions directly to him.

VERY IMPORTANT: We expect all of you to comply with academic integrity. The honor code statement, which has been sent you by email, was prepared to clarify our expectations about the academic integrity principles. Please study the part of this statement related with individual assignments very carefully. If you submit this homework assignment, we will assume that you all read this honor code statement and comprehensively understood its details. Thus, please make sure that you understood it. If you have any questions (any confusions) about the honor code statement, consult with the course instructors.
We will check your codes for plagiarism.