CS 202 – Fundamental Structures of Computer Sciences II
Assignment 2 – BST Implementation

Due Date: July 8, 2012 (Sunday), 18:00

In this homework assignment, you are supposed to implement a C++ program for a course registration database by using binary search tree (BST) structure.

In this course registration database, courses are kept in a binary search tree structure and for each course, you will have a course name and a student binary search tree to keep track of students enrolled to this course.

So, the main BST will keep courses and each course node will have its own BST to keep students enrolled to that particular course.

Each course will be stored according to its course name, such as "cs202", note that course name will always have small letters and no spacing between numbers and letters.

Each student will be stored according to its student id. Also, each student node will have student id, name and surname information. Whereas, each course node will only have course name.

Your system will have the following functionalities:

1. Add Course.
2. Remove Course.
3. List all Courses.
4. Enroll Student.
5. Remove Student from a particular Course.
6. Show class roster of a particular Course in ascending order.
7. Show class roster of a particular Course in descending order.
8. Show all the Courses of a particular Student enrolled in.

Add Course: This function will add a Course node to the BST, **this function will only take course name, which also will be used as key value**. The key comparison can be directly done by '<', '==', and '>' operators. For example: "cs202" < "cs223" results true, because alphabetically cs202 is before cs223. **You can assume that all the letters will be lower case and there will be no space between any letter and number.** If the course is already in the BST then a new entry should **not** be done. This function will take only course name.

Remove Course: This function will remove a Course node from the BST, if the course is not present in the BST, then you need to report that. **You can use both in order successor or predecessor removal strategy.** This function will take only course name.

List all Courses: This function will print all the present courses in the BST structure in ascending order as in the example output.

Enroll Student: This function will add a student node to the student BST of a particular course. Note that there are two levels of BST, one keeps track of courses and each course has its own BST to keep track of students enrolled to that course. **You will use student id as key value for insertion**, if the student is already enrolled to that particular course or that course does **not** exist, it should be reported and further actions should **not** be taken. This function will take four parameters, course name, student id, student name, student surname.

Remove Student: This function will remove a student node from the student BST of a particular course. If the student or course does not exist, it should be reported and further actions should **not** be taken. **You can use both in order successor and predecessor removal strategy.** This function will take only two parameters, course name and student id.

Show class roster of a particular Course in ASCENDING order: This function will list all the students enrolled to a particular course in **Ascending** order, that means student with smallest Id first, the one with largest Id last. It will take only one parameter, which is course name.

Show class roster of a particular Course in DESCENDING order: This function will list all the students enrolled to a particular course in **Descending** order, that means student with largest Id first, the one with smallest Id last. It will take only one parameter, which is course name.

Show Courses of a particular Student: This function will list all the courses that a particular student is enrolled in. It will take only one parameter, student Id.

Below is the required CourseRegistrationDatabase.h file's content, you will prepare the cpp file and submit all the required files in an archive file (zip, tar, tar.gz, rar). Note that the test environment will be dijkstra server, if your code fails to run due to portability issues, you will get zero and will not have a chance to correct your code. You should test your code in dijkstra server before submit.

```cpp
class CourseRegistrationDatabase {
public:

	CourseRegistrationDatabase();
	~CourseRegistrationDatabase();
	void addCourse( const string courseName);
	void removeCourse( const string courseName );
	void listCourses();
	void enrollStudent( const string courseName, const int studentId,
	                                  const string studentName, const string
studentSurname);
	void removeStudent( const string courseName, const int studentId);
	void showClassRosterInAscendingOrder( const string courseName);
	void showClassRosterInDescendingOrder( const string courseName);
	void showCoursesofaStudent ( const int studentId);
};
```

Example Test Code: The below is an example test code, note that the actual test code will be more complicated, will test all the cases. You should also extend this test code and try more complicated cases.

```
#include "CourseRegistrationDatabase.h"
#include <string>
#include <iostream>
using namespace std;

int main()
{
	CourseRegistrationDatabase crd;

	crd.addCourse("cs223");
	crd.addCourse("cs223");
	crd.addCourse("cs201");
	crd.addCourse("cs225");
	crd.addCourse("cs202");
	crd.addCourse("cs101");
	crd.addCourse("cs102");
	crd.addCourse("cs319");
	crd.addCourse("cs315");
	crd.addCourse("cs351");
	crd.addCourse("cs476");
	crd.addCourse("cs224");
	crd.addCourse("cs317");

	cout << endl;
	crd.listCourses();

	cout << endl;
	crd.removeCourse("cs225");
	crd.listCourses();

	cout << endl;
	crd.enrollStudent("cs225", 5, "s5", "sn5");

	cout << endl;
	crd.enrollStudent("cs202", 5, "s5", "sn5");
	crd.enrollStudent("cs202", 5, "s5", "sn5");
	crd.enrollStudent("cs202", 4, "s4", "sn4");
	crd.enrollStudent("cs202", 2, "s2", "sn2");
	crd.enrollStudent("cs202", 3, "s3", "sn3");
	crd.enrollStudent("cs202", 1, "s1", "sn1");
	crd.enrollStudent("cs202", 0, "s0", "sn0");
	crd.enrollStudent("cs202", 7, "s7", "sn7");
	crd.enrollStudent("cs202", 6, "s6", "sn6");
	crd.enrollStudent("cs202", 8, "s8", "sn8");
	crd.enrollStudent("cs202", 9, "s9", "sn9");
	crd.enrollStudent("cs202", 9, "s9", "sn9");

	cout << endl;
	crd.removeStudent("cs202", 4);
	crd.showClassRosterInAscendingOrder("cs202");

	cout << endl;
	crd.removeStudent("cs202", 7);
	crd.showClassRosterInAscendingOrder("cs202");
```

```
        cout << endl;
        crd.removeStudent("cs202", 8);
        crd.showClassRosterInAscendingOrder("cs202");

        cout << endl;
        crd.removeStudent("cs202", 9);
        crd.showClassRosterInAscendingOrder("cs202");

        cout << endl;
        crd.removeStudent("cs202", 6);
        crd.showClassRosterInAscendingOrder("cs202");

        cout << endl;
        crd.showClassRosterInDescendingOrder("cs202");

        cout << endl;
        crd.removeStudent("cs202", 10);

        cout << endl;
        crd.enrollStudent("cs223", 3, "s3", "sn3");
        crd.enrollStudent("cs201", 3, "s3", "sn3");
        crd.enrollStudent("cs224", 3, "s3", "sn3");

        cout << endl;
        crd.enrollStudent("cs319", 7, "s7", "sn7");
        crd.enrollStudent("cs315", 7, "s7", "sn7");
        crd.enrollStudent("cs351", 7, "s7", "sn7");

        cout << endl;
        crd.showCoursesofaStudent(3);
        crd.showCoursesofaStudent(7);
        return 1;
};
```

Example Output: The below is the output of the code given.

```
course: cs223 is added.
cs223 already exists.
course: cs201 is added.
course: cs225 is added.
course: cs202 is added.
course: cs101 is added.
course: cs102 is added.
course: cs319 is added.
course: cs315 is added.
course: cs351 is added.
course: cs476 is added.
course: cs224 is added.
course: cs317 is added.

Listing all courses:
    cs101
    cs102
    cs201
    cs202
    cs223
```

```
        cs224
        cs225
        cs315
        cs317
        cs319
        cs351
        cs476

course: cs225 is removed.
Listing all courses:
        cs101
        cs102
        cs201
        cs202
        cs223
        cs224
        cs315
        cs317
        cs319
        cs351
        cs476

cs225 does not exist.

student: 5 is enrolled to the course: cs202
student: 5 is already enrolled to the course.
student: 4 is enrolled to the course: cs202
student: 2 is enrolled to the course: cs202
student: 3 is enrolled to the course: cs202
student: 1 is enrolled to the course: cs202
student: 0 is enrolled to the course: cs202
student: 7 is enrolled to the course: cs202
student: 6 is enrolled to the course: cs202
student: 8 is enrolled to the course: cs202
student: 9 is enrolled to the course: cs202
student: 9 is already enrolled to the course.

student: 4 is removed from the course: cs202
Listing class roster in ascending order:
        0 - s0 sn0
        1 - s1 sn1
        2 - s2 sn2
        3 - s3 sn3
        5 - s5 sn5
        6 - s6 sn6
        7 - s7 sn7
        8 - s8 sn8
        9 - s9 sn9

student: 7 is removed from the course: cs202
Listing class roster in ascending order:
        0 - s0 sn0
        1 - s1 sn1
        2 - s2 sn2
        3 - s3 sn3
        5 - s5 sn5
        6 - s6 sn6
        8 - s8 sn8
        9 - s9 sn9

student: 8 is removed from the course: cs202
Listing class roster in ascending order:
        0 - s0 sn0
        1 - s1 sn1
        2 - s2 sn2
        3 - s3 sn3
        5 - s5 sn5
        6 - s6 sn6
        9 - s9 sn9

student: 9 is removed from the course: cs202
Listing class roster in ascending order:
        0 - s0 sn0
        1 - s1 sn1
        2 - s2 sn2
        3 - s3 sn3
        5 - s5 sn5
        6 - s6 sn6

student: 6 is removed from the course: cs202
Listing class roster in ascending order:
        0 - s0 sn0
        1 - s1 sn1
        2 - s2 sn2
        3 - s3 sn3
        5 - s5 sn5

Listing class roster in descending order:
        5 - s5 sn5
        3 - s3 sn3
        2 - s2 sn2
        1 - s1 sn1
        0 - s0 sn0

student: 10 is not enrolled to the course: cs202

student: 3 is enrolled to the course: cs223
student: 3 is enrolled to the course: cs201
student: 3 is enrolled to the course: cs224

student: 7 is enrolled to the course: cs319
student: 7 is enrolled to the course: cs315
student: 7 is enrolled to the course: cs351
```

```
student: 3 is enrolled to courses:
    cs201
    cs202
    cs223
    cs224
student: 7 is enrolled to courses:
    cs315
    cs319
    cs351
```

Notes:

1. This assignment is due by 18:00 on Sunday, July 8th, 2012. You should upload your homework to the upload page (http://cs.bilkent.edu.tr/~oalbayrak/cs202/hw2/) before the deadline. This upload page will not be available after July 11th. 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 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.

3. 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 "`CourseRegistrationDatabase`" and your files' name MUST BE "`CourseRegistrationDatabase.h`" and "`CourseRegistrationDatabase.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, do not submit a file that contains a function called "main".

4. 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 will get zero. 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.