Binary Search Trees (BST)

Due Mar 2, 2020 by 11:59pm **Points** 10
Available Feb 24, 2020 at 12am - Mar 2, 2020 at 11:59pm 8 days

This assignment was locked Mar 2, 2020 at 11:59pm.

0. READ THE INSTRUCTIONS BELOW **COMPLETELY AND CAREFULLY** BEFORE PROCEEDING.

0.1 THE CLASS LECTURE ON BST HAS MORE DETAILS ON HOW THEY WORK. READ IT BEFORE PROCEEDING.

Introduction

1. In this assignment you will be implementing

a) a binary search tree (BST) using C++ classes and composition, and

b) a simple database that stores student information using inheritance

2. This time, the declaration for the required classes are provided. These must not be changed, and your implementation must follow them.

3. First, implement the BST (declared in bst.h) and use the unittest_bst() function to test it. DO NOT CHANGE THIS CODE FOR YOUR SUBMISSION, as it will be used to test your code against the answers (with different seed values).

4. Implement the DB and use the unittest_db() function to test it. DO NOT CHANGE THIS CODE FOR YOUR SUBMISSION, as it will be used to test your code against the answers (with different input files).

a) The student ID should be the base class's key member.

b) The student ID should be automatically assigned such that the ID/key should be n if the student is the nth student to join the school. If the student later leaves (i.e., deleted from the BST), the ID does NOT get reassigned to another student. Thus, the student ID of the last student to join the school should reflect the TOTAL number of students that have joined this school since its reception (regardless of whether some have left).

5. Test your code against the provided input and output files.

a) The provided answer for the BST unit test is in "unittest_ans_t100_s100.txt". The s100 refers to the seed of 100 (-s 100), and t100 refers to the number of elements to add to the BST (-t 100).

b) The provided answer for the DB is in "students_1_ans.txt" for the "students_1.txt" input file.

6. Make sure your code has no memory leaks (using valgrind).

7. Your code should work beyond the provided unit tests. That is, even if it does work for all the given tests, if the code has an identifiable bug (i.e., by reading the source code), points WILL be deducted.

For example, if I were to change

unittest_bst(num_test, seed, cout, 5); ->

unittest_bst(num_test, seed, cout, 100);

it should still work.

8. As before, do the homework in your own repo, commit, and **push to Bitbucket**. If you do not push to Bitbucket, the TA and I cannot see the code, and it will be considered a late assignment (i.e., not graded).

Some Rubric (1)

Criteria	Ratings		Pts
unittest_bst 1 Unit test for basic BST using some random -t and -s options	**2 pts** **Full Marks**	**0 pts** **No Marks**	2 pts
unittest_bst 2 Unit test for basic BST using some random -t and -s options	**2 pts** **Full Marks**	**0 pts** **No Marks**	2 pts
unittest_db 1 Unit test for DB using some random set of input students	**2 pts** **Full Marks**	**0 pts** **No Marks**	2 pts
unittest_db 2 Unit test for DB using some random set of input students	**2 pts** **Full Marks**	**0 pts** **No Marks**	2 pts
Comments, Readability, and Understandability	**2 pts** **Full Marks**	**0 pts** **No Marks**	2 pts
			Total Points: 10