BinaryTree Templates, Lab Assignment 13

Re-submit Assignment

Due Nov 29 by 11:59pm **Points** 100 **Submitting** a file upload **File Types** cpp

No, we're *not* actually writing **BinaryTree.h**. BST's are implemented in the STL's **map** and **set**, so we'll just use theirs.

For this assignment use any combination of the C++ STL containers, **array** (C++11), **vector**, **deque**, **stack**, **queue**, **list**, **priority_queue**, **set**, and/or **map**. Do *not* use any of your own templates.

Part 1

College catalogs sometimes list courses that have not been offered in many years, and departments have no intention of offering them again. It's a disservice to students and to the community to continue to list them. For this (and several other reasons) the administration wants a program that will output the last semester in which a course was offered.

Write **DvcScheduleSearch.cpp** to solve the problem. In a loop, prompt the user to enter a course name (like COMSC-210) including a subject code (like COMSC), a dash, and an alphanumeric sequence number. If a user enters uppercase or lowercase X, quit the loop and the program -- be sure to explain this in the prompt. For any other input, look up the course and output the last time that course was offered, nicely labeled (like "COMSC-210 was last offered in Fall 2014"). Or if the course is not found, output something like this: "CS-100 could not be found as far back as the year 2000" because that's as far back as the database goes.

The response to user input should be practically instantaneous -- so there's no time to open and read the TXT file for every inquiry. Also remember that ***Spring comes before Summer comes before Fall***.

Data Structures In Real Life: This solution actually got deployed on **https://web.dvc.edu/coursehistory** (https://web.dvc.edu/coursehistory) . You can use that page to check your own CPP, because the results should match (except that the live page uses the *current* version of the database of course offerings, and your TXT file is an earlier snapshot of that database.)

Part 2

Each section-term combination is supposed to be unique -- that is, there should never be two separate entries with the same section and term, with different course names. It's okay for COMSC-210-8301 in sp2016 to appear more than once -- that's a duplicate and we already skip the extra entries. But it's *NOT* ok to have both COMSC-210-8301 *and* CNT-120-8301 in sp2016. There can be "cross-listed" courses, like ARCHI-126 and ENGIN-126 in Fall 2013 in the same room at the same time with the same instructor (Tumlin, MW 6:00-8:50pm ET-124), but they always have different section numbers, like ARCHI-126-*8349* and ENGIN-126-*8346*.

The DVC Admissions Office suspects that there are at least a few such invalid entries in their database, but their staff is having trouble using Excel to locate them. They heard about what we are studying in COMSC-210, and they want to know if we can help determine if there are any such entries, and what they are.

Write **DvcScheduleCheck.cpp** to solve the problem. Output how many term-section pairs there are with multiple courses associated with them. If there are none, output a message to that effect. But if there are any, output each of them in a format that you think DVC Admissions would understand and find useful.

Data Structures In Real Life: This solution did not get deployed as such. It was developed and used in response to a specific request from DVC Admissions, and only the results were communicated.

Lab Assignment Rubric

Criteria	Ratings							Pts
Fully accurate results, following all specifications **view longer description**	Works the first time. 70.0 pts	Works on the 2nd try 65.0 pts	Works on the 3rd try 60.0 pts	Works after 4 or more tries. 50.0 pts	Doesn't work after 2 weeks. Partial credit. 20.0 pts	Not submitted within two weeks of the due date. 0.0 pts	Work is not original -- appears to be a marked-up copy of the work of another or previous student. 0.0 pts	70.0 pts
Submits all work on time, fully complete if not fully correct. **view longer description**	Submitted on time 20.0 pts	Submitted on time, but one or more files are missing or not correctly named. 16.0 pts	Submitted on time, but with missing identification in one or more submitted CPP or H files. 15.0 pts	Submitted on time but not fully complete. 10.0 pts	Late or wholly incomplete! 0.0 pts			20.0 pts
Well-organized and professional quality code. **view longer description**	Fully meets expectations 10.0 pts	Mostly meets expectations, just needs to be a bit more careful. 8.0 pts	Many areas are well done, but there are a lot of areas that need work. 6.0 pts	Getting there, but needs to be a lot better. 3.0 pts	Needs a lot of work. See the instructor for guidance. 0.0 pts			10.0 pts
								Total Points: 100.0