University of Newcastle
Discipline of Computing and IT
Semester 2, 2017 - SENG1120

Assignment 3

Due using the Blackboard Assignment submission facility:
11:59PM – Friday, 3 November 2017

NOTE: *The important information about submission and code specifics at the end of this assignment specification***.**

In lectures, we have discussed the use of binary trees. This assignment will require the implementation of a binary tree and several algorithms for element insertion, search, removal, and tree traversal.

Assignment Task

Your task in this Assignment is to implement a data structure based on a binary search tree, which will store generic objects, i.e. use templates. The tree must use dynamic node and tree classes – i.e. you can't use arrays.

The type of object used in this assignment is called `Student` and it should store two data values: `string name` and `int grade`.

The data structure should implement functions to add, remove and search for an element, as shown in the lectures – i.e. you must use recursion.

Specific Steps for Implementation

1) Initialize a binary search tree and add the following students and grades **(names should be added from first to last column)**. To find the position to add the objects, you should use `name` to guide the search/insertion process.

Hugo	Jamey	Lidong	Karm	Jianjun
Edouard	Tyson	Aaron	Kundayi	Max
Tobias	Sarah	Prajna	Sebastian	Benoit
Binbin	Trent	Corey	Darcy	Fadzayi
Blake	Matthew	Stephen	Elijah	Jonathan
Kopara	Bryce	Lea	Brandon	Cliff
Cameron	Naneth	Reid	Hadia	Andrew
Jesse	Robert	Radhika	Sean	Jacob
Timothy	Liam	James	David	Luke
Lachlan	Dylan	Mitchell	Ryan	Geoffrey

87	40	37	78	79
50	42	47	96	57
99	55	85	58	69
53	75	12	32	54
20	34	30	87	40
88	61	25	88	17
49	56	87	19	46
31	72	7	28	6
8	26	40	21	57
97	73	36	27	72

2) Print out all names and grades using the inorder traversal, i.e the output should be:

`(name1, grade1) (name2, grade2) … (nameN, gradeN)`

3) Print out how many students had a grade < 50, by recursively traversing the tree, i.e. the output should be "`FF: XX`"

4) Print out the average grade of the class, also by recursively traversing the tree, i.e. the output should be "`Average: XX`"

5) Delete all students who had a grade < 50. When a node is deleted, it should be replaced so that the tree that maintains the binary search consistency. *This is a complex procedure and you should refer to the textbook, pp. 1373 to 1378, to understand all the possibilities for node deletion.* We will also talk about this during the classes.

6) Redo steps 2, 3 and 4 with the tree after the deletion process.

7) Input a name from the user and search for it in the tree. If it is found, print out the name and the grade as `(name, grade)`. If it is not found, print out "Name not found".

Your submission should be made using the Assignments section of the course Blackboard site. **Incorrectly submitted assignments will not be marked. Assignments that do not use the specified class names will not be further marked.** You should provide all your `.h`, `.cpp` and `.template` files, and a `Makefile`. Also, if necessary, provide a `readme.txt` file containing any instructions for the marker. Each program file should have a proper header section including your name, course and student number, and your code should be properly documented.

Remember that your code should compile and run correctly using Cygwin. There should be no segmentation faults or memory leaks during or after the execution of the program.

Compress all your files, including the cover sheet, into a single .zip file and submit it in by clicking in a link that I will create in the Assignments section on Blackboard especially for that.

Late submissions are subject to the rules specified in the Course Outline. Finally, a completed Assignment Cover Sheet should accompany your submission.

This assignment is worth 15 marks of your final result for the course.