CS3D5A – Assignment 3 - Trees and Search

Anton Gerdelan gerdela@scss.tcd.ie and Peter Lavin peter.lavin@scss.tcd.ie

Due midnight **29 November**.
Weight: 1/4 of CA grade. 10% of module.

1 Deliverables

- Any *C* or *C++* source code files used {.c, .cpp, .h, .hpp} with your name and date in comments at the top of every file.
- Do not include project settings files or compiled files.
- Work is individual. Do not copy-paste code from others or the web.
- No written report for this assignment.

2 Instructions

The goal of this assignment is to build a program with the data structure and functions of a typical **binary search tree**. Start by writing a suitable structure to represent a node that contains a `char` as the data record. Create a node pointer in your main function to represent the root of a tree. Next implement the following functions, and test them from `main()`:

1. `void tree_insert( Tree_Node* root, char data );`

   Creates a new node in the tree in the appropriate position for a binary search tree.

2. `Tree_Node* tree_search( Tree_Node* root, char data );`

   If a value exists in the tree, return a pointer to that node, otherwise return NULL.

3. `void tree_print_sorted( Tree_Node* root );`

   Traverse the tree, printing the data held in every node, in smallest-to-greatest sorted order.

4. `void tree_delete( Tree_Node* root );`

   Delete every node in the tree without creating a memory leak.

5. A function or functions that will generate a **balanced** binary search tree from an array of characters. The function should call your insertion function in the correct order, which it should determine. Use an array containing the following values as test data, but your function should work with any similar array: X, Z, C, B, A, Y.

Comment all functions and key sections with 1-3 lines to show your intent, and any assumptions you made.

3 Grading Scheme

- Your program must compile.
- 2 marks per function, assuming correct functionality, readability, clear comments, and no bugs.

4 Suggestions

- If you need help getting started ask us.
- Make sure that you know how binary search trees are supposed to work and be structured before you start programming. Draw diagrams of the expected result on paper to help you test your code.
- The tasks are arranged here in order of difficulty - start at the top, and reserve more time for later tasks.
- Probably the easiest approach is to design each function as a recursive process. Think about the correct traversal layout for each one - in-order, post-order, or pre-order.
- Adding print statements to your functions and following these steps against your on-paper diagrams should help you understand problems.
- Turn on compiler warnings. With GCC or Clang this is the `-Wall` flag. Debug build is the `-g` flag.
- Draw diagrams to help understand confusing pointer sequences.
- By all means ask for help with programming problems or tricky tasks. Pointers and recursion are confusing even for experts. You can paste confusing or broken snippets of your code into the discussion boards to ask for advice. Don't wait until the last few days, so that we have time to help you.