## CSCI 2720

### Project 2: Binary Search Tree

### Project Due: July 29th before 11:50pm

You will implement a C++ binary search tree (bst) of nodes that contain integer values. Your program will build a binary search tree based on a sequence of insert and delete operations given in an input file. The bst should not allow duplicate integers to be inserted. If a duplicate integer is attempted to be inserted into the bst, then that insert operation should be ignored. Also, if an integer is attempted to be deleted from the bst that is not in the bst, then that delete operation should be ignored. Furthermore, when deleting a node ***p*** that has a right child, ***p*** should be swapped with the node ***q*** that contains the minimal value of ***p***’s right subtree, and ***p*** should be deleted afterwards. You may use any of the following data structures in the C++ STL: vector, array, list, stack, queue, and deque. Your program must construct the bst as shown in the examples provided on eLC, and it must follow the same formatting of the output as shown in the examples.

Additional Requirements

1. Your program must include a makefile, called makefile (do not add a file extension), that compiles your bst source files into an executable object called bst with a compile directive. The makefile should also have a run directive and clean directive.
2. Your program must process an input file, whose name is given in argv[1], that contains a sequence of insert and delete operations and their operands. The file will be formatted with lines that contain at most one operation of insert or delete and its integer operand. The first string on a line containing an operation will contain the string insert or delete followed by a single space followed by an integer. Lines that do not contain an insert or delete operation should be ignored, and any characters after an integer operand should also be ignored. Your program must build its bst by doing each operation, in order, from the top of the input file until the end of the input file is reached. Examples of input test files and their outputs can be found on eLC, and your bst should be constructed in the same manner as the examples. Study the examples provided to determine how insertions and deletions are handled.
3. After building the bst based on all of operations provided in its input list, your program must display the following, in the order specified, to standard output
   1. Number of nodes in the bst
   2. Height of the bst
   3. Pre-order traversal of the bst
   4. In-order traversal of the bst
   5. Post-order traversal of the bst
   6. Level-order traversal of the bst

The output of your program must match the format of the output examples provided on eLC.

1. [Extra Credit Tree Visualization, optional] After displaying the information in part 3, display the entire tree structure to standard output. The root should be at the top of the tree visualization. The spacing on each line of your display must be done such that it looks like a neatly organized binary search tree when all levels of the bst are printed out together. The display should clearly show that the outputted integers form a binary search tree, and the children of each node should be easy to visualize. This extra credit is worth up to 20 points depending on how well the visualization is implemented. The instructor and TA will not help with the extra credit.

### Other useful files attached

The following files are located on eLC, and they contain sample input files and their corresponding output files for a correctly implemented bst. Your program must build the bst and display its output as shown in the examples files provided.

test0.txt and its output file, output0.txt

test1.txt and its output file, output1.txt

test2.txt and its output file, output2.txt

etc …

**Testing**

You must test that your program compiles and runs correctly on nike using /usr/bin/g++. You are responsible for testing your program with our test cases and additional test cases that you create on your own. Thoroughly and rigorously test, test, and re-test your program before submitting it to ensure that it works correctly on any test cases that we may use for grading.

### Submission Procedure

In your home directory on nike, make a subdirectory named "proj2" and put copies of your source files in it, as well as the header files and the makefile. Also, include a readme file with your name, any information you'd like the grader to see such as how to compile and run your code.

.

Once you’ve LOGGED ONTO nike and you are in your home directory, execute the command line "submit proj2 cs2720". After submitting, verify that a file beginning with “rec” was created in your submitted directory. This “rec” file is your receipt of submission. If a “rec” file was not created, then the submission was unsuccessful, and you’ll need to try resubmitting. If resubmitting does not work, zip or tar your directory and email it to the instructor and TA before the deadline.

**Late Policy**

This is the last project, so please submit whatever you have done, NOT later than Tuesday July 31 11:59pm, else TA wouldn’t have enough time to grade and submit your project evaluation for final course grade.

**Grading**

|  |  |
| --- | --- |
| Programming style, commented code, following directions,  working makefile, and readme included | 20 points |
| Program correctness based on various test cases | 80 points |
| Total | 100 points |
| Extra credit | 20 points |