Homework 6 – Binary Search Tree

DUE: December 5 by 11:59:59 PM **Assigned**: November 13

Background

A binary search tree is a specialized tree data structure that auto-sorts as items are inserted in or deleted from the tree. Typically, insert, search, and delete operations run in O(lg n) time, but the worst case is O(n). See Chapter 12 in textbook.

Problem

You will write a templated Binary Search Tree (BST) class.

Assignment Requirements

- The name of your executable shall be `hw06`
- Your source code, object files, and executable shall be organized
  - See the *Preparing and Submitting* section
- Download the header file from Blackboard
  - **DO NOT:**
    - Change the name of the class
    - Change the name of any function
  - **DO:**
    - Apply `const` where it should be applied
    - Add any extra private functions and data that may be beneficial

Class Requirements

- Use the `Album` class available on Blackboard (will be handy for testing `emplace()`
- Apply `const` on all applicable return types and member functions
- See the BST.hpp file for requirements of a function
- You are free to implement the class "as you see fit"
  - Obviously, the requirements must still be met
  - This relates more to your specific algorithms, and how they go about doing their jobs (erase can be done as shown in class, or recursively, etc.)
  - It also relates to the helper functions you decide to use

main.cpp Requirements

- Test your class thoroughly
- Test files will be made available during the last week of the assignment period

Hints

- IMPORTANT: Because we are dealing with a template class, testing is just a little different

    - The linter will not catch as much as it usually does, this is due to the runtime nature of templates
    - A function won't really show that it's broken until you try using it
- Thorough testing is extremely important
    - Does your function work on an empty tree?
    - Does your function work on a "normal" tree?
    - Does your function work on the two possible "linked list" trees?
    - Does your function work on a jagged ("lightning bolt") tree?
- All the exceptions required to be thrown can be found in the <stdexcept> library

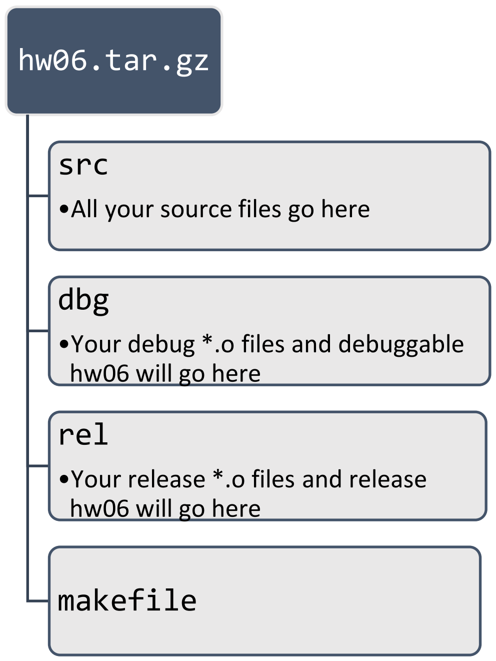
Reminders

- Include a makefile!
- Be sure to include a comment block at the top of every file with the required information
    - Refer to the General Homework Requirements handout on Blackboard
- Provide meaningful comments
    - If you think a comment is redundant, it probably is
    - If you think a comment is helpful, it probably is
    - Remember that you are writing comments for other programmers, not people who know nothing (obligatory Jon Snow) about coding
        - Emphasize 'why' over 'what'
- There will be no extensions

--- CONTINUED ON NEXT PAGE ---

Preparing and Submitting

- Your code must be able to compile and run on the EECS Linux Cluster
  - You are responsible for testing your code
  - "But it runs fine on my machine!" will **not** earn you any points
- Submit **ONLY** source code files and your makefile
- You will submit a zipped tarball (*.tar.gz) of your assignment
  - The structure of your assignment shall follow the example below:


hw06.tar.gz
src
•All your source files go here
dbg
•Your debug *.o files and debuggable hw06 will go here
rel
•Your release *.o files and release hw06 will go here
makefile

NOTE: You do not have to include the dbg and rel folders in your tarball, BUT if you don't, your makefile should be able to generate them

- Your submission must include a makefile
- Submit ONLY code files and a makefile as a tarball
  - From inside your project directory, run the following command:

```
$ tar -czvf hw05.tar.gz src makefile
```

- Design your makefile so that the executable file is called hw05
- Homework submission will be handled exclusively through the handin tool in the Linux Cluster. You may submit your homework using the following command:

```
~cs400/bin/handin 6 hw06.tar.gz
```