CSE 100 PA 1: Binary Search Trees

This document will guide you through your first C++ programming assignment, where you will implement a BST as a C++ class template that conforms to the basic conventions of the C++ Standard Template Library. This PA will offer more guidance than usual. Be sure to read this assignment, the starter files, and the referenced resources. Start early!

Final Submission Due: Friday August 5th, 2016, 11:59PM.

Required Final Submission Files:

BST.hpp, BSTIterator.hpp, BSTNode.hpp, main.cpp, README

Provided Starter Files:

actors.txt, actors_sorted.txt, BST.hpp, BSTIterator.hpp, BSTNode.hpp, Makefile, test_BST.cpp, main.cpp

Table of Contents:

Approaching CSE 100 PAs

Read the entire assignment early. Start planning the code early. Submit often and hours before the due time.

Why read the assignment early? Because you'll give yourself more time to "digest" the information that is outlined here.

Why start planning the code early? Because as we progress through the PAs, you'll find that you'll be given more room for your own individual implementation, which requires more of your creativity. You will also be more likely to have access to tutors and the professor in their hours. Get into a good habit before it's too late.

Why submit often? Because we will be using Vocareum and ieng6 to run our testers on your submissions. You will know whether you are missing files, whether it's compiling, and whether it has basic implementation issues that need to be fixed. Therefore, the best way for you to do the assignment is to use the ieng6 lab machines or SSH into the ieng6 server in order to match our compiler. Your code must compile on the ieng6 and Vocareum systems.

Part 0 – Academic Integrity Form and Pre-Survey

In order for your assignments to be graded, you must understand and agree to the course rules for the integrity of scholarship.

Academic Integrity Form Here

You may choose to work independently or in pair programming, but you **may not** split the work up between partners. The full guidelines are here.

Part 1 – How to get the Starter Code, Partner in Vocareum

Log into your account in the lab or through SSH/VNCViewer.

Lab Account Information:

https://sdacs.ucsd.edu/~icc/index.php

The starter files are located in:

```
/home/linux/ieng6/cs100v/public/pa1
```

Copy these files into your own folder in your account.

If you don't have a cs100v account and are using your own account, you need to

```
prep cs100v
```

Part 2 – Implementing a BST in C++

Your first task is the *understand the code*. Do not attempt to start coding until you have a firm understanding of what the provided code does and what you need to do with it. Nearly all the required functionality is specified in the code and comments, so not only do you need to understand the assignment, but you need to understand the structure of the provided code.

We highly recommend test-driven development. Make sure you can understand the provided test file, test_BST.cpp, which partially tests some aspects of the BST class. Then, add your own tests before you start implementing any code of your own. Note that it is up to you to be thorough with your tests; our provided testers will only test basic functionality.

The provided Makefile will compile test_BST.cpp and create the bst executable file. Do not edit the Makefile.

In this assignment, you will be implementing the following BST functions and features:

- Insert: given a reference of a Data item, insert a copy of the Data into the BST.
- Find: look for a Data item in the BST.
- deleteAll: clears the BST. Already implemented, study this algorithm (provided for you as part of the starter code).
- Empty: tells whether the BST is empty.
- Inorder: Prints the Data of each node using in-order ascending traversal.
- Size: number of elements in the BST.
- Height: height of the BST.
- The iterator pattern: way to traverse a BST by command, another method of debugging.
- A search program: way to look for Data inside a BST (more in Part 4)

The starter code for these are in the .hpp files. Find them and then provide your full implementation of the functions marked with //TODO. You should remove nothing from the code except for the //TODO comments when you finished writing the functions. Details are in the comments.

You may add your own variables and/or functions, but these must be in a **"private:**" section to indicate that they are part of your design but **not meant for public or protected access for other classes.** More details about the functions are provided in the starter code.

Assume that clearing memory leaks, fixing segfaults, and having good style are part of the correctness portion of the assignment. Refer to the course website for more info.

Part 3 – Using a BST in C++

In main.cpp, you need to implement the main function which does the following:

1. When the user runs this main program using the command line ./main <filename>, the program will open the formatted .txt file. (This step is included in the starter code.)
2. Read the .txt file with various actor names line by line. Load them into the BST. Each line of the .txt file is a unique actor name.
3. Output the BST's size.
4. Output the BST's height.
5. Prompt the user to enter an actor's name.
6. Output the result of the search. (Whether the name was found.)
7. Continue to prompt the user to enter an actor name until the user quits by typing "n" and pressing Enter.

Main.cpp gives detailed comments on the specific formatting of the input, output, and prompts. **You must follow these formats to gain all the points.**

Part 4 – README

In a new README file with no file extensions, answer the following questions. You may search online for help, but you must write the answers in your own words. Do not copy and paste from anywhere. These questions should help you understand the starter code and some C++ concepts.

1. In the file BSTNode.hpp, you can see that the signature of the constructor is
   ```
   BSTNode (const Data & d) : data (d)
   ```
   Explain why the parameter d must be a constant lvalue reference, rather than just a non-constant l-value reference. (Assume we do not have another overloaded constructor.)
2. Notice that the BSTIterator class overloads both the pre-increment and the post-increment operators.
   a. Why does the post-increment operator take an argument?
   b. Why does the post increment operator return a copy of the BSTIterator while the pre-increment operator return a reference?

Unsure about any parts of the assignment? We highly recommend you to come to tutor and office hours so that you can talk about the assignment with a tutor or the professor. See the tutoring calendar for the schedule of hours.

If you have not read the starter code, now would be a great time to do so.

Part 5 - Studying and Exploring

Possible exam concept: Explore the height of your BST with different inputs. We provide a sorted actors file (actors_sorted.txt) and an unsorted actors file (actors.txt). Run main on both of the files. Explain the relationship with the number of actors in the file and the height of the tree, and why this occurs. Do not submit anything from this part, but it would help you to talk to classmates and staff or discuss on Piazza on this topic.

Part 6 – Submitting on Vocareum

Log into Vocareum and select the PA you want to submit to. Press “My Work”. In the page you entered, click the “Upload” button to your left. Choose the files you want to submit.

A script will run. **Make sure** that your submission at least:

1. Has all the correct names.
2. Compiles.
3. Runs without issues with the tester.

Fix any errors that arise from Steps 1-3. The testers guarantee only basic correctness (not complete correctness; you must write your own testers.)

Only one of the programmers in a partnership need to submit, but both should verify that the submission works for both programmers.

You can submit once, ten times, any number of times before the deadline and only your last submission will be graded. If you submit within 24 hours after the deadline, you will be charged a slip day. Slip days cannot stack, so there will be no submissions allowed after 24 hours.

Part 7 – Grading Summary

This PA is worth 25 points:

- README questions → 3 points
- Style → 2 points
- Correctness → 20 points.

Memory leaks will result in deductions. Each test case that results in a segfault will receive a 0. Compile and link errors will cause the entire assignment to receive a 0. Valgrind and GDB are highly recommended debugging tools to use.

Part 8 – Other Notes and Tips

1. **The `virtual` keyword exists in order to implement more sophisticated, high-performance search tree structures.** If you are not sure what a virtual function is in C++, assuming you understand the concepts of polymorphism and dynamic dispatch from Java, a simple Google search on "virtual function C++" should get you to some understandable resources. You can also ask the course staff.

2. **The '==' operator might not be overloaded**. You will have to implement the insert and find methods with just the '<' operator for data comparison. The Standard C++ STL Library uses equivalence instead of equality to compare values for ordered sequences (in our case, BSTs).

3. **Back up your code often.** After every successful step in writing code, make a backup. If you are in lab, bring a flash drive. If you are SSHing, use some form of SCP. If you are using Remote Desktop like vncgnome, use some form of SCP or private cloud storage. **Make sure your code is accessible ONLY by you (and your partner if applicable).**

4. **There is a multitude of ways to work remotely.** ieng6 can open Firefox by typing into terminal `firefox`. This is useful if you are SSHing with X11 forwarding. You can also use the scp command or WinSCP to transfer files between your computer and ieng6. VNCViewer is also another good way to work remotely. *But working directly in B250 labs is still the best option*.

5. **If you're using Vim or gVim, editing your .vimrc file will save you time and energy.** Here's a pre-built one with instructions. This was made to work with Linux; if you want to use it on Macvim or Windows gVim, you'll need to modify the font.