24783 Advanced Engineering Computation: Problem Set 5

In this problem set, you write three programs: (1) binary-tree rebalancing, (2) self-balancing binary bree, and (3) unit test for the binary-tree code.

(*) In the following instruction (and in all of the course materials), substitute your Andrew ID for where you see *yourAndrewId*.

START EARLY!

1 Check Out or Update Base Code and Libraries

Please make sure you have up-to-date libraries and course files before starting an assignment.

If you have not done working-directory set up as described in the first assignment (like in case you need to work from a different computer), please see Problem Set 1 and set up the working directory.

I assume you created the working directory called *24783* under you home directory and you checked out your Git repository in there.

Home directory is typically *C:\Users\username* in Windows, */Users/username* in macOS, and */home/username* in Linux, where *username* is the user name in your local computer.

First, open command-line (Developer PowerShell or Terminal), and move to your working directory by typing:

```
cd ~/24783
```

You need to check out (or clone) Git repositories once. If you have not checked out yet, do the following:

```
git clone https://yourAndrewId@ramennoodle.me.cmu.edu/Bonobo.Git.Server/course_files.git
git clone https://yourAndrewId@ramennoodle.me.cmu.edu/Bonobo.Git.Server/yourAndrewId.git
```

You need to replace ”yourAndrewId” with your Andrew ID. You'll be asked to type in credentials.

Also we are going to use two additional repositories:

```
git clone https://github.com/captainys/MMLPlayer.git
git clone https://github.com/captainys/public.git
```

If you are successful, you should have the following directory structure under your home directory.

```
Your User Directory
 └─(Other files and directories)
 │
 └── 24783
      ├── course_files
      └── yourAndrewID
```

If you already have checked out these repositories (most likely you did for Problem Set 1), you need to update (or git pull) in those repositories. By change directory to the location where you checked out repositries and then type:

```
git pull
```

To update all four repositories, you can type the following commands in a sequence:

```
cd ~/24783/course_files
git pull
cd ~/24783/yourAndrewID
git pull
cd ~/24783/public
git pull
cd ~/24783/MMLPlayer
git pull
```

2 Copy Base Code and Add to Git's Control

The base code is in the ps5 sub-directory in course_files. First, copy this ps5 sub-directory to your directory by "cp -r" command.

You can write ps5_1 and ps5_2 programs (graphical) from the program in bintreeviz sub-directory. Therefore, rename bintreeviz to ps5_1, and then make a copy of ps5_1 and name it ps5_2. Change-directory to your ps5 sub-directory, and then you can do so by typing:

```
mv bintreeviz ps5_1
cp -r ps5_1 ps5_2
```

ps5_3 is a console application. No graphics. Therefore, make sub-directory called ps5_3, and then make an empty main.cpp (maybe just one line like int main()) in there for now.

Your directory tree should look like the following.

```
Your User Directory
 └── 24783
      ├── course_files
      ├── public
      └── yourAndrewID
           └── ps5
                ├── (PDF files etc.)
                ├── bintreelib
                ├── ps5_1
                │    └── main.cpp
                ├── ps5_2
                │    └── main.cpp
                └── ps5_3
                     └── main.cpp
```

Once done, change directory to ps5, and type "git add *".

3 Make CMake Scripts

Write a top-level CMakeLists.txt in ps5 sub-directory. Top-level CMakeLists.txt should include ps5_1, ps5_2, pst_3, and bintreelib. Make sure to enable C++11. Also use enable_testing().

In each sub-directory, write CMakeList.txt files. ps5_1 and ps5_2 are graphical programs. Therefore, you need to use MACOSX_BUNDLE. Also link fssimplewindow, ysbitmapfont, and bintreelib libraries. ps5_3 is a console application for testing binary-tree class. Therefore it does not require fssimplewindow and ysbitmapfont libraries. It requires bintree library only.

bintreelib (in bintreelib sub-directory) is a binary-tree library. Everything is written in bintree.h file. However, CMake wants at least one .cpp file to define a library. Use bintree.cpp to keep CMake happy.

The target names must be ps5_1, ps5_2, ps5_3, and bintreelib (same as the directory name).

Then "git add" the CMakeLists.txt files you made.

4 Binary-Tree Re-Balancing (45 points)

In PS5-1, you implement the binary-tree re-balancing algorithm presented by Quentin F. Stout and Bette L. Warren in 1986. http://web.eecs.umich.edu/ qstout/pap/CACM86.pdf

1. Download and read the paper.
2. Implement right rotation function in BinaryTree class in bintreelib.
3. In main.cpp of ps5_1, when the user presses R key, apply right rotation to the tree node under the mouse cursor.
4. In binarytree.h, add a member function called TreeToVine, which does tree-to-vine transformation as described in the paper.
5. In binarytree.h, add a member function called VineToTree, which does vine-to-tree transformation as described in the paper.
6. In main.cpp of ps5_1, call TreeToVine and VineToTree functions when the user presses T and V keys respectively. The tree should be re-balanced when the user presses T and then V.
7. In main.cpp of ps5_1, insert a new random number to the tree (between 0 and 99) when the user presses the SPACE key. Make sure you can re-balance the tree after adding a bunch of numbers.
8. In main.cpp of ps5_2, change the rendering so that all lines are black, and labels of the nodes are drawn black if the left and right heights are equal, red if right heavy, or blue if left heavy.

5 AVL Tree (45 points)

In PS5-2, you implement a type of self-balancing binary-tree called AVL-tree. Read the explanation in the lecture note for more details about the algorithm.

1. In BinaryTree class, add a flag (bool) called autoRebalancing as a public member variable. The default value of this flag must be false.

2. Write a protected member function called Rebalance, which takes a node pointer or a node handle and applies the re-balancing algorithm of the AVL-tree. This function must check and re-balance the given node and all the up-stream nodes.
3. In Insert function of the BinaryTree class, apply re-balancing if autoRebalancing flag is true.
4. In Delete function of the BinaryTree class, apply re-balancing if autoRebalancing flag is true.
5. In main.cpp of ps5_2, turn on autoRebalancing when the program starts (or in the constructor of ApplicationMain class). Do it before the first number is inserted.
6. In main.cpp of ps5_2, insert a random number between 0 and 99 when the user presses the SPACE key. Make sure your tree is kept balanced.
7. In main.cpp of ps5_2, change the rendering so that all lines are black, and labels of the nodes are drawn black if the left and right heights are equal, red if right heavy, or blue if left heavy. (Same as ps5_1)

6 Unit Test (10 points)

In ps5_3, implement Last and FindPrev functions in BinaryTree class and write a program for testing the two functions.

Last and FindPrev are symmetric to First and FindNext functions respectively.

Think how you can test the two functions. The main function of the test program must return 0 if the test is successful, or non-zero (like 1) if unsuccessful.

Also add a unit test (called *ps5test*) by using add_test and enable_testing commands in CMakeLists.txt. enable_testing must be in the top-level CMakeLists.txt. add_test must be in CMakeLists.txt in ps5_3 sub-directory.

The name of the unit test must be *ps5test*.

7 Test Your Code on the Compiler Server

Test your source files (.cpp and .h files) on the compiler server. Some assignment may not require .h files. You do not have to test files that you don't make modifications. The files you need to test are the ones you write or modify.

We have four compiler servers:

- http://freefood1.andrew.cmu.edu:24780
- http://freefood2.andrew.cmu.edu:24780
- http://freefood3.andrew.cmu.edu:24780
- http://freefood4.andrew.cmu.edu:24780

Make sure you don't see red lines when you select your files and hit "Comiple" button on the server.

We have multiple servers to make it less likely that all of them need to shut down for maintenance. If do not have to test on all of the servers. You need to make sure that your code passes on one of the servers.

8 Submit

Lastly, you need to submit using git. What you need to do are two things: (1) add files to git's control, and then (2) send to the git server.

8.1 Add Files to git's control

If you haven't done yet, add all the files under ps5 subdirectory (excluding build files if you made a build directory under ps5). To do so, type:

```
git add ~/24783/yourAndrewID/ps5
```

This command will add ps5 directory and all files under the subdirectories.

8.2 Send to the Git Server

In Git, sending files to the server is a two-step process. The first step is local commit. You can do it by:

```
git commit -m "Problem Set 5 solution"
```

The message can be anything, but it is recommended to type something meaningful, at least you can see what changes you made to your repository.

Local commit is just local. Git server does not know about any local commit unless the commit is sent (or pushed) to the server. To do so, type:

```
git push
```

Make sure to do it in the CMU network. If you are working from home (probably most likely), use VPN to connect to the CMU network.

You can re-submit (commit and push) your solution as many times as you want with no penalty before the submission due.

9 Verification

It is recommended to clone your repository to a different location and make sure that all of your files have been sent to the Git server.

You can do the following:

```
cd ~
mkdir 24783Verify
cd 24783Verify
git clone https://yourAndrewID@ramennoodle.me.cmu.edu/Bonobo.Git.Server/yourAndrewId.git
```

Once you made sure all the files have been submitted, you can delete files and directories under 24783Verify directory.

10 AVL Tree Performance Competition

We measure time that your AVL-tree class takes for adding 20,000,000 random numbers, and then deletes all even-numbers. Top-3 submissions will get 20 bonus points. Next 2 will get 10 bonus points. Will be measured on the same computer. For each person, 3 measurements will be taken and the fastest one will be used.

As long as you keep the public members of your binary-tree class unchanged, and as long as you are writing (and using) your own AVL tree algorithm, you can make changes to the part that is not exposed to the outside of the class.

The submission will be disqualified if:

- I cannot compile the test code with your binary-tree library. Common reason is due to a naming convention of library, file, class, or function name,
- If the numbers are not ordered, or
- If the tree is not balanced.

If your code beats my AVL-tree class that I'm using for my research work, you'll get 3% bonus toward the final grade. It is not the fastest code in the world. Some parts are written in a conservative fashion. You have a good chance of beating it!