EECS 293
Software Craftsmanship
2016 Fall Semester

Programming Assignment 7

Due at the beginning of your discussion session on

October 17-21, 2016

Reading

Read Section 19.6 in Code Complete and the Quick Reference on Routine Names on blackboard.

Grading Guidelines

Starting with Programming Assignment 7, an automatic C (or less) is triggered by improperly named routines.

An automatic C (or less) is also triggered by

- Any routine with complexity greater than 4, or by
- Any piece of code that is essentially repeated.

Programming

In this programming assignment, you will implement the data structure for which you wrote the pseudo-code.

First, make all the changes to your pseudo-code as per the review in your discussion section.

Second, read "Using git in EECS 293" posted on blackboard under Course Documents>git, and create the project repository on eecslinab6. You will need the public-private key pair that you created for a previous homework assignment.

Third, create a new repository "opponents.git". You should place all of your homework artifacts in the opponents.git repository.

Fourth, set up a development environment that mirrors the one on the server. This way, your compilation and testing will give predictable and consistent results both on your machine and on the server, as well as on the machine of your discussion leader. The server environment is described on blackboard under Course Documents > Server Set-Up.

Fifth, create an ant file to build and test your project. The testexample repository provides an example of a build file, which you can modify to match the structure of your project. Conversely, Eclipse build files are overly specific to your machine and should be avoided. Your ant build file (or Makefile) should contain at least:

- A target called "build" that compiles your code (Ruby: compiles your code but does not run it).
- A target called "test" to run the test cases.

Sixth, implement the data structure according to the pseudo-code programming process. Follow your revised pseudo-code faithfully, even if you can think of additional improvements. You will probably need to implement some methods for error handling. However, since your code is primarily for hypothetical future use as a component in a larger project, your code should contain only simple stubs for error handling.

Seventh, make sure during development you make small regular commits. As you push your project, the git server will attempt to build toward the ant or make targets, and will give you feedback on the outcome. You can then adjust your source code and build file (or Makefile) to ensure proper compilation of your code. You are not allowed to commit and push standard libraries, such as hamcrest-core, or files that can be automatically generated (class, jar, html, and testing and coverage report). Make sure to use a .gitignore file, such as the one provided in testexample, that tells git to skip files that should not be tracked.

Finally, when you are finished with your homework tag your release and push that tag to your git repository on eecslinab6.

```
$ git tag -a pa7
$ git push --tags
```

Blackboard Resources

The Course Documents contain:

- *PPP: Huffman Coding*, an example of the pseudo-code programming process, starting from the pseudo-code for creating a Huffman tree and ending with its Java implementation,
- *git*, a folder with instructions for "Using Git in EECS 293", and pointers to several free books on git,
- *Server Set-Up* gives instructions on creating a development environment that mirrors the one on the server,
- *Apache Ant*, a link to the download page and documentation of Apache Ant,
- *JUnit*, a folder with:
    - *JUnit* points to the JUnit page, with installation instructions.
    - *Junit ant task*, a reference to the JUnit task in ant, and
    - *Ant + Junit* explains how to set up a build file that contains a JUnit "test" target.

General Considerations

These classes may contain as many auxiliary private methods as you see fit, and additional helper classes may be defined.

You should write JUnit tests to make sure that your primary methods work as intended. However, we will revisit testing in the next assignment, so extensive testing is not yet recommended. Similarly, your code should have a reasonable number of comments, but documentation is going to be the topic of a future assignment. As a general guideline at this stage of the course, comments and tests should be similar to those accepted in EECS 132.

Discussion Guidelines

The first part of the class discussion is on git: the discussion leader will pull your changes from your repository on eecslinab6. You will then give the class a brief overview of your development process using the commit history stored in git. The aim is to have relatively small self-contained commits with descriptive commit messages.

The bulk of the discussion will focus on the pseudo-code programming process (Chapter 9): appropriateness and completeness of pseudo-code for rapid implementation, pseudo-code comments in final code, etc. For the sole purpose of comparing with the previous assignment, leave all pseudo-code comments in your code.