CMPSC 112
Introduction to Computer Science II
Fall 2016

Laboratory Assignment Eight: Performance Evaluation of a Security Program

Introduction

In the past laboratory assignments and classroom discussions we have investigated: (i) algorithms, (ii) data structures, (iii) the Java programming language, (iv) tools for software engineering, and (v) the analytical and empirical evaluation of algorithms. In this laboratory assignment, you will specify, design, and implement your own system that solves an interesting problem in the field of computer security. In particular, you will implement and evaluate a program that analyzes and improves passwords. Your program will perform various checks to determine if the user's password is secure. If one or more of these checks fail, then you will improve the password until all of them pass. You will intuitively prove the worst-case time complexity of an algorithm in your program and then conduct a doubling experiment to confirm the correctness of your suggested time complexity.

Review Your Textbook

To do well on this laboratory assignment, you should review the content in several sections of the textbook. Please study Sections 1.8 and 1.9 to learn more about packages and import statements in Java and the correct way to engineer software. Next, you should carefully review all of Chapter 2, paying particularly close attention to Sections 2.1, 2.3, and 2.5 and their treatment of object-oriented principles. You should also study all of Chapter 4, focusing on the examples, featured on pages 176 and 177, that reveal how to prove the worst-case time complexity of an algorithm. Finally, if your password checking and improvement system requires the use of recursion, then you should review the material in Chapter 5. Please see the instructor with all of your questions!

Features of the Computer Security Program

Instead of relying on code that the course instructor provides to you, this assignment requires you to start the project "from scratch", relying on your prior laboratory assignments as a source of examples and inspiration. To start, you should implement a program that performs password checking. It should ensure that the user's input password contains at least one capital and lowercase letter, a symbol that is not alphanumeric, and that it is longer than eight characters in length. Your program should also improve a user's password if it does not meet one of the aforementioned criteria. For instance, if the provided password does not contain a capital letter, then you should include at least one capital letter at some location in the password. If the password is not long enough, then the program should add appropriate characters to it until the length is suitable.

The program that you implement should adhere to the principles of object-oriented design. In addition, the source code of the program should be organized into one or more packages and all of the code should be automatically compiled and managed through an Ant build system. Whenever it is necessary to do so, the program should correctly employ the principles of object-oriented programming, correctly using polymorphism, inheritance, and encapsulation and any relevant Java language features such as exceptions, casting, and generics. Finally, the code should be carefully commented using the JavaDoc standard. That is, each Java class should have a "header" comment

and all methods should have their own comment as well. Finally, all of the most important lines in the program should also have comments that clearly explain how the source code contributes a main feature of the system. Please see the instructor if you have questions about these requirements.

Conducting Experiments to Evaluate Efficiency

After you have finished implementing your password checking and improvement program, you should pick one algorithm in your system (e.g., the algorithm that checks for the existence of a capital letter) and intuitively prove the worst-case time complexity of this method, following the examples on pages 176 and 177 as an example. To confirm the correctness of your final time complexity, you should conduct a doubling experiment — either by using your own source code or by fully integrating a new example into the ExpOse tool — and then report on your results in a comprehensive report that contains table(s) of data and a full-featured discussion of the table(s).

Carefully Review the Honor Code

The Academic Honor Program that governs the entire academic program at Allegheny College is described in the Allegheny Academic Bulletin. The Honor Program applies to all work that is submitted for academic credit or to meet non-credit requirements for graduation at Allegheny College. This includes all work assigned for this class (e.g., examinations, laboratory assignments, and the final project). All students who have enrolled in the College will work under the Honor Program.

It is understood that an important part of the learning process in any course, and particularly one in computer science, derives from thoughtful discussions with teachers and fellow students. Such dialogue is encouraged. However, it is necessary to distinguish carefully between the student who discusses the principles underlying a problem with others and the student who produces assignments that are identical to, or merely variations on, someone else's work. While it is acceptable for students in this class to discuss their programs, data sets, and reports with their classmates, deliverables that are nearly identical to the work of others will be taken as evidence of violating the Honor Code.

Summary of the Required Deliverables

This assignment invites you to submit a signed and printed version of the following deliverables:

1. Using diagram(s), an explanation of all of the packages and classes in your program.
2. The final version of the commented source code for your computer security program.
3. A comprehensive written report that fully explains the results of your experimental study.
4. A reflective commentary on the challenges that you faced when completing this assignment.

Along with turning in a printed version of these deliverables, you should ensure that everything is also available in the Git repository that is named according to the convention `cs112F2016-<your user name>`. Additionally, students are encouraged to create a GitHub repository to host the final version of this project. If you decide to use the ExpOse tool to conduct the doubling experiment to study the performance of your system, then you should make sure that you "fork" this project's repository on GitHub and issue a pull request that initiates the review of your new example. While students are encouraged to leverage past laboratory assignments for useful examples, you should carefully cite the source code of any project that you developed as part of a team.