Software Testing Mini Campaign (Individual)

200 Points (+ Bounty Points)

Problem Statement: Data Reconciliation

Description: Consider a CSV file that stores a list of records (e.g., records of bank accounts). You are required to write a software program that reads two such CSV files, compares records stored in these CSV files row by row against a unique combination and records all mismatches as exceptions. Finally, the software program generates another csv file listing the exceptions.

Example: Consider customer files that contain customer id, customer account number, currency, account type (e.g., savings/current) and available balance. Compare available balance against a unique combination of customer id, customer account number, account type and currency. Generate a CSV file with records from both the files and corresponding to the mismatched amount (i.e., balance) for the unique combination.

Scope of Implementation and Testing: The assignment will be carried out between Week 8 and Week 13, over a period of six weeks, and will be judged based on the following guidelines:

1. You will use a version control system (preferably github or likewise) for this project. Please make sure not to submit anything in eDimension.
2. **[10 Points, Week 8]** You are required to draw a use-case diagram for the assignment. Augment the use-case diagram with one or two misuse cases. You may assume that a malicious user can only interact through the inputs given to your program.
3. **[20 Points, Week 8]** You are required to use Java for this implementation. Additionally, you must implement it from scratch. Other than standard I/O, file systems etc., usage of libraries and plugins are not allowed (when in doubt, feel free to consult with Sudipta). Last but not the least, include a proper and self-contained README file for someone to run your code independently. Here is a good example of documenting a README file.
4. **[15 Points (equivalence class) + 15 points (boundary value), Week 9]** Work out an equivalence class partitioning and boundary value analysis for blackbox testing of your program. Explain all the equivalence classes, examples of boundary/middle values in each equivalence class and the rationale behind your choices.
5. **[30 Points (unit testing) + 30 points (system testing), Week 10]** Perform unit testing and system level testing on your code. Since you are using Java, use the JUnit framework to write the unit test cases, as discussed in the class/offline lecture videos. Try to write parametric tests as much as possible. Your system-level test design should align with your boundary value analysis and equivalence class partitioning. White box testing is optional for this point.
6. **[10 Points, Week 12]** You are required to follow good coding practices. For example, handling all exceptions, checking for potential escapes (e.g. passing object reference outside class) should be handled in your code. Additionally, your code must be well documented and have a README file for others to run your code independently.
7. **[20 Points (fuzzer design) + 20 Points (fuzzing impl. and results), Week 12]** Fuzz your implementation. Recall that one of the primary objectives of Fuzzing is to find

irregular inputs that lead to unexpected behavior. To this end, you need to find your fuzzing target and the inputs that need to be fuzzed. Write a fuzzer to validate the robustness of your implementation. You can use any language to implement your fuzzer and use any of the ideas discussed in the class (e.g., random fuzzer, mutation-based, feedback-based etc.). It is important that you write the fuzzer yourself and do not use any off-the-shelf tools.

8. **[30 Points + variable bounty, Week 13]** You will test the implementation of your project group members (at least three different implementations excluding yours). A group member X will share with another group member only the code implementing the problem statement and no test code written by X would be shared. Your objective is to use any testing techniques (blackbox, whitebox, fault-based, fuzzing) to discover bugs in your group members' implementations. Again, you are not allowed to use off-the-shelf tools to test your group members' implementation. All test code to validate your group members' implementation needs to be saved (even though the tests do not find any bugs). For each new (unique) bug found in your group member's code, you will get 5 bonus points. It is the responsibility of each student to provide a proper README file so that someone else can test their code without much trouble and without much communication between group members.

Submission Guidelines: You will not submit anything in eDimension. Instead, you will add myself and the teaching assistants in your github (or similar repositories). Do not make your repository public at any point during the course. We will evaluate the deliverables at the end of each week based on the commits. For example, deliverables in Week 8 will be evaluated based on all commits available until 11.59pm, 10th July. Similar rules will apply for other weeks. More detailed instructions are as follows:

- **Week 8** [Document and Code]: Use case diagram should be complete. All Java (or similar) files should be complete in terms of implementing the functionality and (in the bare minimum) the sample test files should pass. **Deadline 10th July, 11.59pm**.
- **Week 9** [Document]: A report on your boundary value analysis and equivalence class partitioning should be complete. **Deadline 17th July, 11.59pm**.
- **Week 10** [Code]: All (Junit) test files conducting unit and system level testing should be complete. **Deadline 24th July, 11.59pm**.
- **Week 12** [Code]: A refactored/bug fixed version of the code. The fuzzer code is complete and properly documented for running by a third-party. The implementations of others are also distributed this week for independent testing by group members. **Deadline 7th Aug, 11.59pm**.
- **Week 13** [Document and Code]: All new tests written for the third party implementation should be complete. A report to briefly explain the bugs found in other's implementation should be complete. Each new bug found in another implementation will have a bounty (in terms of bonus marks: 5 marks / unique bug). **Deadline 14th Aug, 11.59pm**.

Sample Tests: In the Google drive folder for the assignment, three sample CSV files are included. You may like to use it for some preliminary testing. Note that you are required to write your own tests to validate the project code. Thus, the included sample CSV files should only be taken as examples for preliminary testing. You should perform a thorough testing as listed out in points 5 and 7 of implementation and testing scope.

For *sample_file_1.csv* and *sample_file_2.csv*, most (if not all) entries should return exceptions, as the files are generated via two independent random processes.

sample_file_1.csv and *sample_file_3.csv* contain two exceptions if we compare unique combination of ID, account no, currency and account type: specifically, for account ID99 and ID198 (balance does not match). However, for a unique combination of ID, account no and account type, there is an additional exception: for ID298 (currency does not match).

Github handle of teaching assistants and Sudipta: sudiptac (Sudipta Chattopadhyay). TBA for the teaching assistants.