Assignment 1: Test Suite Analysis and Improvement

Course: EECS 4313 - Software Engineering Testing
12.5 marks
Due date: Fri Feb 20, 5pm

Objective
Evaluate and improve unit test coverage of a set of standalone simple Java classes, provided in JavaAlgs project in GitHub, using coverage analysis, LLM-generated tests, and manual refinement.

Instructions

1. **Clone the repository at** JavaAlgs

2. **Initial Coverage Analysis (2.5 marks)**

   - Run the existing unit tests per program and calculate the following:
     - Statement and branch coverage (use tools like, JaCoCo)
     - Report one function per project where “all condition-decision coverage” is different than “all decision (all branch) coverage” and calculate their all condition-decision coverage with the existing test cases, *manually*.
       - Hint: the two metrics are different if the decision’s logic is not an atomic condition. For example: (x>0) is atomic but (x>0) && (y<0) is not.
   - Submit:
     - Pre-improvement coverage **report,** including all raw logs (copy-paste or screenshots) as well as a table summarizing the statement and branch coverage per program
     - The “all condition-decision coverage” analysis and comparison with “all branch coverage” in the reported functions.

3. **LLM-Generated Test Improvement (3 marks)**

   - Use an LLM (e.g., ChatGPT, Gemini, DeepSeek) to generate extra test cases for every program.
     - Target low-coverage areas. Try different prompts and context engineering to get the best out of your LLM, but do not reveal the correct answers for example by giving away the assertion you are looking for.
   - Ensure:
     - Generated tests are runnable and relevant (do not include irrelevant/bloated tests).

    - LLM-generated tests must improve the coverage (anywhere statement or branch are less than 100%). Try to maximize your coverage.
- Submit:
    - All LLM-generated test code as a zip file
    - Include three examples in the report, where LLM created interesting test cases
    - Include the detail about which LLM(s) you used, what was/were the prompts and if you had to make multiple runs include all. For example if you did prompt engineering show what you start with and what was the issue and how you resolved it.
    - Post-LLM coverage report (table showing before/after coverage improvement).

4. LLM Test Analysis (2.5 marks)

- Critically evaluate the LLM's output:
    - Good Practices: E.g., meaningful assertions, edge-case coverage.
    - Bad Practices: E.g., redundant tests, incorrect mocks.
    - Report at least 3 strength and 3 weakness patterns in the generated tests. Provide concrete example per pattern.
- For any weak or poorly generated test, fix the test manually
- Submit:
    - An analysis report.
    - The fixed test cases. In the zip file

5. Manual Test Improvement (2.5 marks)

- Manually add tests to increase one coverage metric in at least 2 programs, compared to the existing and LLM-generated tests.
- Manually add more tests to achieve 100% coverage (same coverage metrics) in at least 2 programs that were not 100% before
- Submit:
    - Manual test code (annotate improvements). Both in the zip file and in the report
    - Post-manual coverage report (table showing the three states of improvement).

6. Final Reflection (2 marks)
- Reflect on:
    - Challenges in selecting/running tests.
    - LLM limitations vs. human intuition.
    - What LLMs are missing + ideas on how to improve LLMs in TestGen.

    - Insights into test adequacy and quality.
- Submit:
    - Reflection as part of the final report.

Grading Scheme

Criteria	**Marks**	**Requirements**
Initial Coverage	2.5	Metrics reported.
LLM Test Improvement	3	Tests runnable, coverage boosted.
LLM Test Analysis	2.5	≥3 str./wk., Fixed properly, depth of examples.
Manual Test Improvement	2.5	≥2 units improved per metrics, ≥2 at 100%
Final Reflection	2	Clear reflection, actionable insights.
Penalties	**up to -5**	**Execution errors on tests, tool misuse**
	-up to 5	**Poorly written report**

Submission

Zip file for test codes and a report document.
The report document contains the following:

- Coverage reports (initial, post-LLM, post-manual) with the tables on the metrics.
- Test code wherever asked explicitly in the details instruction above
- LLM generated test analysis and final reflection.