# Lab Assignment 5

**Course:** CS202 Software Tools and Techniques for CSE
**Lab Topic:** Code Coverage Analysis and Test Generation
**Date:** 6th February 2025

**Objective**
This lab aims to analyze and measure different types of code coverage and generate unit test cases using automated testing tools. Students will work with a given dataset containing Python programs and evaluate various testing metrics, including:

- Line (statement) Coverage
- Branch Coverage
- Function Coverage

Students will learn to use automated test generation tools and understand their advantages and disadvantages.

**Learning Outcomes**
By the end of this lab, students will be able to:

- ✓ Understand and differentiate various types of code coverage.
- ✓ Use automated tools to measure coverage on a given dataset of Python programs.
- ✓ Write/Generate effective unit tests to maximize coverage.
- ✓ Visualize coverage reports and analyse test effectiveness.

**Lab Requirements**

- ➢ Operating System: Windows/Linux/MacOS (**it is strongly advised to use a VM, preferably SET-IITGN-VM to work on this assignment**)
- ➢ Programming Language: Python 3.10 only (setup a venv for this)
- ➢ Required Tools:
  - pytest (for running tests)
  - **pytest-cov** (for line/branch coverage analysis)
  - **pytest-func-cov** (for function coverage analysis)
  - **coverage** (for detailed coverage metrics)
  - **pynguin** (for automated unit test generation)[1]
  - Some other tools (**genhtml**, **lcov**) introduced in Lecture 5

**Lab Activities:**

1. Clone the keon/algorithms repository to your local and setup the required dependencies in a different environment. Report the current commit hash in the documentation for this lab.
2. Configure `pytest`, `pytest-cov`, `coverage`, etc. to only inspect this repository for dynamic analysis.

[1] https://stackoverflow.com/questions/69409598/how-to-set-the-environment-variable-pynguin-danger-aware-for-pynguin

Note: Please reach out to the TAs for any queries/issues.

3. Execute existing test cases (test suite A) provided with the repository, record the coverage metrics using appropriate tools already configured in the previous step.
4. Analyze whether the test suite A covers ALL lines, branches, functions in this repository. Generate visualizations and record them.
5. Generate unit test cases (call this test suite B) for the files that are not covered completely, using **`pynguin`**. In this step, your task is to improve code coverage to the extent possible. Hence, try to generate as many test cases as possible including test cases wild enough to crash the program![2]
6. Compare the effectiveness of test-suite B versus A for each selected repository, in terms of code coverage, etc.
7. Report whether the generated test cases revealed any uncovered scenario[3].

**Resources**

- Lecture 5 slides
- `pynguin` (https://www.pynguin.eu)
- `coverage` (https://coverage.readthedocs.io/en/latest)
- `pytest` (https://docs.pytest.org/en/7.4.x/index.html)
- `pytest-cov` (https://github.com/pytest-dev/pytest-cov)
- `pytest-func-cov` (https://pypi.org/project/pytest-func-cov)

[2] This way you can perform some adversarial attack.

[3] The definition and interpretation of a scenario is left as an exercise.

Note: Please reach out to the TAs for any queries/issues.