Summary and Discussion

Team Members: Jayanth Reddy Gaddam(1002123569)
Jorge Catano (1002149092)

This project provided an opportunity to apply control flow testing principles to a program of moderate size, specifically the “Printtokens.java” class. Through the practical implementation of concepts covered in this course, we reinforced our understanding of software testing techniques, including control flow graph (CFG) generation, edge coverage testing, and fault identification. Below is a summary of the process and a discussion of key insights gained.

Project Summary

1. Control Flow Graph Generation:
   - CFGs were created for each method in the “Printtokens.java” program.
   - The CFGs excluded the code in catch clauses as instructed, focusing on the primary control flow paths within each method.

2. Unit Testing:
   - Each method, other than the `main` method, was subjected to edge coverage testing.
   - Test paths were systematically selected to cover all edges in the CFGs. Any infeasible test paths encountered were replaced with new, feasible paths.
   - Test cases, including test data and expected outputs, were documented for each selected path.

3. Program Testing:
   - The `main` method was tested to achieve edge coverage across the entire program.
   - This included integrating the CFGs of individual methods to form a comprehensive control flow graph for the program.
   - Edge coverage reports were generated for all executed tests.

4. Fault Identification and Correction:

- Seeded faults were located by analyzing test outputs against expected results.
- Faults included logic errors, missing conditions, and incorrect outputs, which were corrected based on testing feedback.
- Fault documentation was prepared to explain the identified errors and their corresponding fixes.

5. JUnit Test Implementation:
- Test cases were implemented in JUnit to automate the execution and validation of test paths.
- The tests provided detailed reports on edge coverage and highlighted areas of potential improvement.

6. Deliverables:
- All project artifacts, including CFGs, test cases, source code, and code coverage reports, were organized and stored in a structured GitHub repository under a "deliverables" folder for accessibility and reproducibility.

Discussion

This project demonstrated to us the importance of conducting systematic testing in order to develop reliable software. Creating the Control Flow Graphs (CFGs) was very useful to visualize the structure of the program and identify sources of problems and strange behaviors. Despite the goal of achieving 100% extended coverage we had to compromise with some unrealistic paths which made us learn ways of attaining the greatest coverage without compromising.

Incorporating JUnit was easier, reduced the manual work and gave precise results than doing it personally. Seeded faults were used for debugging to enhance our abilities for identifying and correcting mistakes and to enhance our overall knowledge of faults. Maintaining proper organization of our project not only benefited from the aspect of being able to collaborate seamlessly but also from the added advantage of having easy access to the work done and in a format that could be reviewed. In conclusion, this project proves that proper testing and the basis of good teamwork is important in creating software.