Data table of experiment results

Team Testers:

- **Rohith Reddy K - 862466402**
- **Ravi Teja N - 862468077**

Statement Coverage:

Tcas.c

Test Case Priortization	Number of Test cases in the test suite	Faults Exposed (Total - 41)	Fault Versions Exposed
Random Priortization	6	11	["v2", "v5", "v12", "v14", "v15", "v27", "v34", "v36", "v37", "v38", "v40"]
Total coverage	4	9	["v1", "v7", "v17", "v23", "v28", "v30", "v35", "v36", "v40"]
Additional coverage	4	9	["v1", "v7", "v17", "v23", "v28", "v30", "v35", "v36", "v40"]

Totinfo.c

Test Case Priortization	Number of Test cases in the test suite	Faults Exposed (Total - 23)	Fault Versions Exposed
Random Priortization	7	10	["v1", "v7", "v8", "v11", "v13", "v15", "v16", "v18", "v20", "v21"]
Total coverage	5	12	["v1", "v5", "v7", "v8", "v9", "v10", "v11", "v12", "v13", "v15", "v16", "v20"]
Additional coverage	5	12	["v1", "v5", "v7", "v8", "v9", "v10", "v11", "v12", "v13", "v15", "v16", "v20"]

Schedule.c

Test Case Prioritization	Number of Test cases in the test suite	Faults Exposed (Total - 9)	Fault Versions Exposed
Random Prioritization	7	1	[“v9”]
Total coverage	3	2	["v2", "v9"]
Additional coverage	3	2	["v2", "v9"]

Schedule2.c

Test Case Prioritization	Number of Test cases in the test suite	Faults Exposed (Total - 9)	Fault Versions Exposed
Random Prioritization	4	1	["v1"]
Total coverage	1	3	["v1", "v8", "v9"]
Additional coverage	1	3	["v1", "v8", "v9"]

Printtokens.c

Test Case Prioritization	Number of Test cases in the test suite	Faults Exposed (Total - 7)	Fault Versions Exposed

Random Prioritization	15	3	["v3", "v5", "v6"]
Total coverage	6	6	["v2", "v3", "v4", "v5", "v6", "v7"]
Additional coverage	5	5	["v2", "v3", "v4", "v5", "v6"]

Printtokens2.c

Test Case Prioritization	Number of Test cases in the test suite	Faults Exposed (Total - 9)	Fault Versions Exposed
Random Prioritization	11	5	["v1", "v2", "v5", "v6", "v8"]
Total coverage	5	8	["v1", "v2", "v3", "v4", "v5", "v6", "v7", "v8"]
Additional coverage	4	7	["v1", "v2", "v3", "v4", "v5", "v6", "v7"]

Replace.c

Test Case Prioritization	Number of Test cases in the test suite	Faults Exposed (Total - 31)	Fault Versions Exposed
Random Prioritization	16	9	["v5", "v12", "v14", "v17", "v18", "v20", "v23", "v27", "v31"]
Total coverage	13	10	["v1", "v2", "v5", "v8", "v12", "v17", "v20", "v23", "v26", "v27"]
Additional coverage	9	6	["v1", "v2", "v5", "v8", "v12", "v27"]

Branch:

Tcas.c

Test Case Prioritization	Number of Test cases in the test suite	Faults Exposed (Total - 41)	Fault Versions Exposed
Random Prioritization	13	11	["v1", "v14", "v16", "v23", "v28", "v30", "v33", "v35", "v36", "v37", "v40"]
Total coverage	13	13	["v2", "v14", "v18", "v22", "v23", "v28", "v29", "v30", "v33", "v35", "v36", "v37", "v40"]
Additional coverage	11	13	["v1", "v7", "v17", "v22", "v23", "v28", "v29", "v30", "v33", "v35", "v36", "v37", "v40"]

Totinfo.c

Test Case Prioritization	Number of Test cases in the test suite	Faults Exposed (Total - 23)	Fault Versions Exposed
Random Prioritization	6	17	["v1", "v4", "v5", "v6", "v7", "v8", "v9", "v11", "v13", "v15", "v16", "v17", "v18", "v19", "v20", "v21", "v23"]
Total coverage	5	13	["v1", "v5", "v7", "v8", "v9", "v10", "v11", "v12", "v13", "v15", "v16", "v18", "v20"]
Additional coverage	5	13	["v1", "v5", "v7", "v8", "v9", "v10", "v11", "v12", "v13", "v15", "v16", "v18", "v20"]

Schedule.c

Test Case Prioritization	Number of Test cases in the test suite	Faults Exposed (Total - 9)	Fault Versions Exposed
Random Prioritization	12	3	["v3", "v5", "v9"]
Total coverage	8	3	["v2", "v5", "v9"]
Additional coverage	7	5	["v1", "v2", "v5", "v6", "v9"]

Schedule2.c

Test Case Prioritization	Number of Test cases in the test suite	Faults Exposed (Total - 9)	Fault Versions Exposed
Random Prioritization	11	4	["v3", "v6", "v8", "v9"]
Total coverage	7	5	["v2", "v3", "v7", "v8", "v9"]
Additional coverage	5	2	["v8", "v9"]

Printtokens.c

Test Case Prioritization	Number of Test cases in the test suite	Faults Exposed (Total - 7)	Versions Exposed

Random Prioritization	14	3	["v3", "v5", "v6"]
Total coverage	7	5	["v2", "v3", "v4", "v5", "v6"]
Additional coverage	6	4	["v2", "v3", "v5", "v6"]

Printtokens2.c

Test Case Prioritization	Number of Test cases in the test suite	Faults Exposed (Total - 9)	Fault Versions Exposed
Random Prioritization	17	5	["v1", "v2", "v4", "v5", "v6"]
Total coverage	6	9	["v1", "v2", "v3", "v4", "v5", "v6", "v7", "v8", "v9"]
Additional coverage	4	7	["v1", "v2", "v3", "v4", "v5", "v6", "v7"]

Replace.c

Test Case Prioritization	Number of Test cases in the test suite	Faults Exposed (Total - 31)	Fault Versions Exposed
Random Prioritization	28	20	["v3", "v4", "v5", "v7", "v8", "v9", "v10", "v11", "v12", "v13", "v14", "v15", "v16", "v18", "v24", "v26", "v27", "v29", "v30", "v31"]
Total coverage	21	19	["v1", "v2", "v4", "v5", "v7", "v8", "v12", "v13", "v14", "v16", "v17", "v20", "v23", "v24", "v26", "v27", "v28", "v29", "v30"]

Additional coverage	11	16	["v1", "v2", "v3", "v4", "v5", "v8", "v13", "v14", "v17", "v20", "v23", "v24", "v27", "v28", "v29", "v30"]

Original Test Suite:

Benchmark Program	**Number of Test Cases**	**Number of Faulty Versions**	**Fault Versions Exposed**
Printtokens.c	4072	7	All versions exposed
Printokens2.c	4057	9	All versions exposed
Replace.c	5542	31	All versions exposed
Schedule.c	2634	9	All versions exposed
Schedule2.c	2679	9	All versions exposed
Tcas.c	1590	41	All versions exposed
Totinfo.c	1026	23	21 out of 23 faults were exposed by the test suite ["v1", "v2", "v4", "v5", "v6", "v7", "v8", "v9", "v10", "v11", "v12", "v13", "v15", "v16", "v17", "v18", "v19", "v20", "v21", "v22", "v23"]

Observations:

1. For Universe.txt all the benchmark programs except Totinfo exposed all the fault version, Incase of Totinfo only 21 fault versions are exposed out of 23
2. Branch coverage is more nuanced which is able expose more faults than statement coverage
3. All though there are huge number of testcases for each benchmark program only **<~1%** could bring 100% statement and branch coverage
4. Although there are test suites which give 100% branch/statement coverage they fail to expose all the fault version
5. The size of test suite which gives 100% branch/statement coverage is very small compared to the original test suite

6. Branch Coverage has more test cases in the test suite when compared to Statement Coverage since branch coverage checks every conditional statement which doesn't happen in case of statement coverage
7. The count of Fault versions exposed with the generated test suites is comparatively less when than that of Universe.txt
8. Branch Coverage is able to expose more faults and is more effective when compared with statement coverage
9. In our case the way we handled Additional coverage prioritization has less testcases in its test suites when compared to Total coverage prioritization
10. For Random prioritization the results (i.e. Test suites & Fault versions exposed) vary significantly with each run due to randomness
11. In most of our cases Total prioritization was able expose more faults than Additional prioritization
12. The size of test suite in our Random prioritization observations is greater than the size of test suites of other two prioritization methods
13. From the project we can distinctly tell the size of test suites doesn't matter for good coverage and fault exposure
14. Branch coverages can do a efficient job of thorough evaluation of programs logical paths and also goes deeper in program execution
15. Although In case of Schedule2.c benchmark program universe.txt has 2679 testcases only one testcase was able to generate 100% Statement coverage in Total and Additional Coverage Prioritization methods

Learnings:

- The project was really interesting we were able to gain some invaluable insights how we can achieve different coverages we were able develop our own parser with the understanding we gained and we had good learning curve
- We came to know more about the available tools in the market and gcov is one such amazing tool
- We went through few of the research work for the project which helped improve our understanding towards test coverage

Note: For most of our queries/issues during our project we had, have been solved by going through Stack overflow threads and also able getting more clarity by professor's help during office hours

ChatGPT / LLMs:

We didn’t use any LLM to write a part of code in our project but we used LLM to understand few nuances in python like running subprocess, creating files and also understanding how gcov works for us

After going through the above results, we were able to do the parsing of gcov file and interpreting the results on our own