Experimental Results:

Size of the test suite:

	tcas	totinfo	schedule	schedule2	printtokens	printtokens2	replace
branch-random	13	9	12	9	15	19	22
branch-total	13	5	8	7	7	6	21
branch-additional	11	5	7	5	6	4	11
statement-random	5	8	6	2	12	14	14
statement-total	4	5	3	1	6	4	13
statement-additional	4	5	3	1	5	4	9

Number of faults exposed:

	tcas	totinfo	schedule	schedule2	printtokens	printtokens2	replace
branch-random	8	14	2	5	4	7	15
branch-total	13	13	3	5	5	7	19
branch-additional	13	13	5	2	4	6	13
statement-random	6	13	0	3	4	6	12
statement-total	9	12	2	3	6	6	10
statement-additional	9	12	4	3	4	5	4
baseline_suite	41	21	9	9	7	9	31

Observations:

More test cases in the branch test suite than in the statement test suite. Random has the most number of test cases. Additional Coverage Prioritization maximizes coverage with fewer test cases.

Branch coverage performs better than statement coverage. Total Coverage prioritization is the best test case prioritization technique. Random prioritization generates the most unstable result which can provide a good and awful test suite.

In summary, Random prioritization, while the simplest method, yields the most variable results; it can occasionally lead to produce a great result but is inherently unstable and unpredictable. Additional Coverage prioritization proves to be efficient as it maximizes coverage with a smaller number of test cases by focusing on incremental coverage benefits. The Total Coverage prioritization technique, which orders test cases by the total number of unique statements or branches covered, emerges as the superior method, likely due to its comprehensive approach to maximizing code coverage.

ChatGPT / LLMs Usage:

We use Chatgpt and Copliot to generate a basic code structure and write some basic functions, such as an argument code, opening a file, and reading from a list.