Homework #1: Manual Test-Case Generation

Due date: 2025-05-04 (23:59)

In the field of software engineering, generating test cases for a target program is a common yet important task. Through this assignment, we want each student to gain hands-on experience in test-case generation.

The goal of this homework is to **manually generate test-cases that maximize code coverage (e.g., the number of covered branches**), which is a widely used metric for evaluating the quality of the test-cases. Let's use the example in Figure 1 to understand how branch coverage is measured. Generally, the number of branches in a program is twice the number of if (or while) statements. Since there are two if statements in the example, the total number of branches is 4. We only need two test cases to cover all the branches in Figure 1. For example, the first test case, where x = 90 and y = 25, will cover the two true branches. The second test case, where x = 0 and y = 0, will cover the two false branches. Together, these two test cases successfully cover all four branches. In our homework, we try to generate test-cases that increase the number of covered branches for a real-world open-source C program.

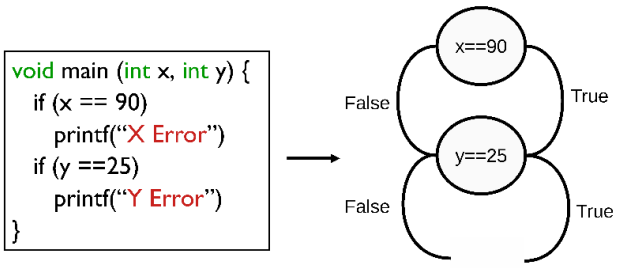


Figure 1 Examples for the number of covered branches

The target program that you will test in this homework is **'m4'**, a macro processing tool that reads input, performs text substitutions based on macros, and outputs the result. Your goal is to execute the m4 program with manually written test cases to maximize the number of branches covered during execution. The more branches your test cases cover, the higher your homework score will be. To measure branch coverage, we will use **gcov**, a widely used coverage analysis tool. It will automatically calculate how many branches of the program were executed with your test cases. To help you design more effective test cases for m4, we also provide a brief guide that explains how the program works and which features you can test to increase coverage.

You may submit up to 30 test cases in total (1 ≤ number of test cases ≤ 30). The number of branches covered by your test cases may vary, and your score will be based on how many branches are successfully covered. Your task is to write test cases for the **m4** program. Each test case should be a single command-line input (examples in example_testcases.txt), and all of your test cases must be written in studentid_testcases.txt. We will provide detailed instructions on downloading the homework files and running the m4 program. Please follow those instructions carefully. If you have any questions, feel free to contact the TA via email or iCampus message.

Version & Environment

- Benchmark: m4-1.4.19
- Evaluation Environment: **Linux Ubuntu-22.04 LTS**

Grading

- Test cases will be evaluated **three times**, and your final coverage score will be the **average** of those three runs.
- Coverage rates will be sorted in **descending order**, and scores will be assigned in groups of 10 students, with the top group receiving 10 points and each subsequent group receiving one point less.
- If your test cases **cause the program to crash**, you will receive an **additional 2 points** regardless of your ranking.
- Each test case must finish execution within **3 seconds**; otherwise, the command will be considered **timed out** and **ignored** during evaluation..
- Since the number of covered branches can vary across Ubuntu versions, the evaluation will be conducted **based on the environment described in this document**.
- Each test case must be written as a single-line command and must not exceed **300 characters**.
- There is no restriction on how the command is constructed (e.g., using echo, pipeline, or input files), but you may only use .m4 files included in the original source code.
- The m4 program must run once per test-case. (not allowed commands like ./m4 .. && ./m4)
- Cases that modify environment variables using ‘export’ or similar methods are not permitted.

Submission

- just submit **studentid_testcases.txt** and upload it to i-campus (change **studentid** to your id).
- Due date: 2025-05-04, 23:59 (No late submission.)

** Caution: If Windows newline “\r” is in testcases, they return errors. Do not write your testcases file on Windows OS.

Instructions

1. Build a benchmark

First, download the *.zip* file from i-campus and extract the *.zip* file.

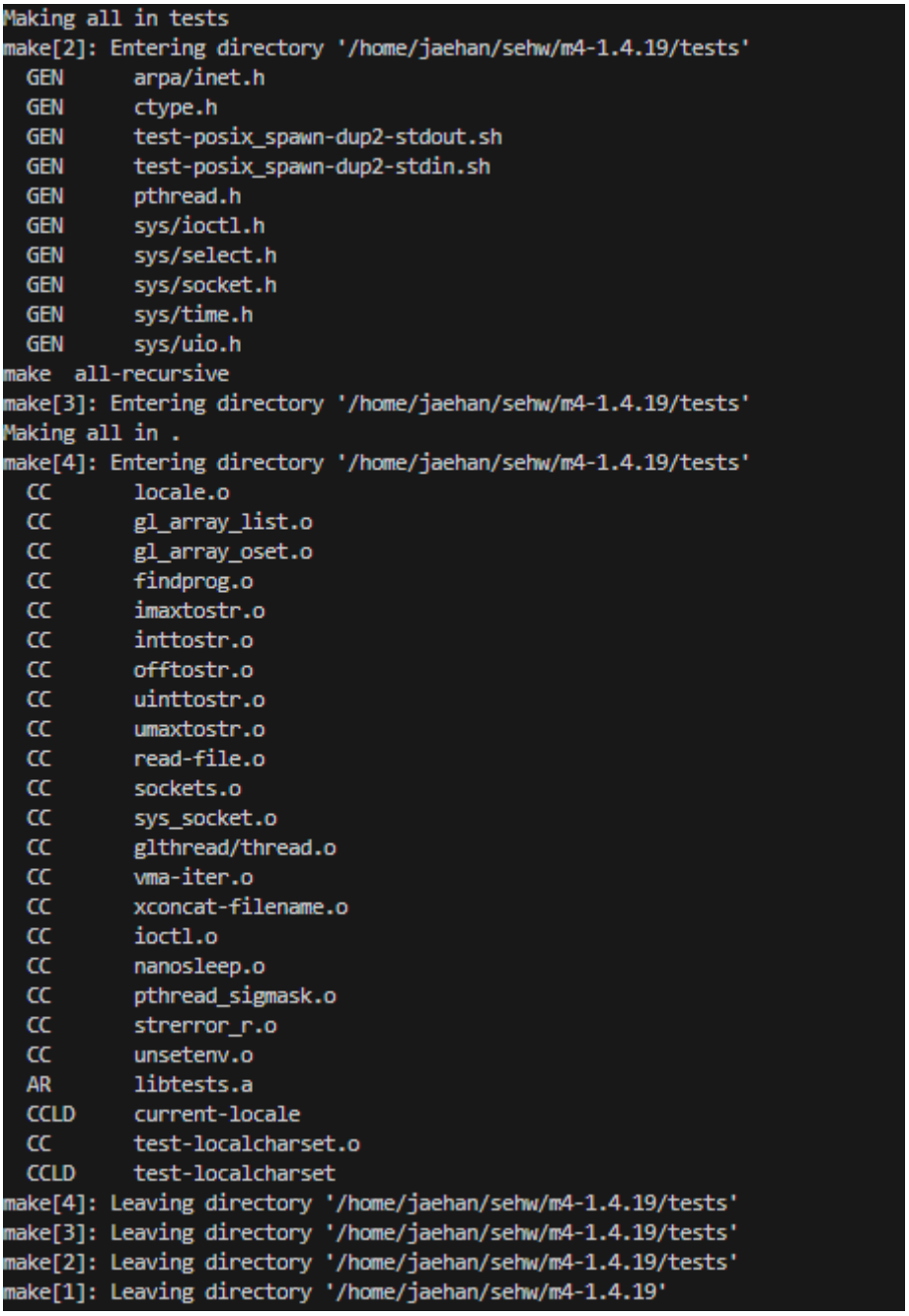
```
$ unzip 2025s_hw1.zip
```

After the given *.zip* file extraction,

```
$ cd 2025s_hw1
$ python3 script.py --build
```

** "--build" option builds m4.

If the build succeeds, the terminal will display the screen shown below.



2. Make test-cases

Open studentid_testcases.txt and write testcases one per line.

(In that text file, there will be two examples.)

3. Check the total coverage of your testcases

```
$ cd ~/2024f_hw1/
$ python3 script.py --testcase_file studentid_testcases.txt
```

** The way script.py works **

1. If you give ***studentid_testcases.txt*** as input, ***script.py*** will run all the commands in the ***studentid_testcases.txt*** file.

   e.g. ./m4 --help

   e.g. ./m4 --version

2. After a binary file is executed (by each testcase), *.gcda* files are automatically created for source codes in ***2024s_hw1/m4-1.4.19.***

   (cannot read *.gcda* file with "cat" or "vi" command)

3. Then it executes gcov and gcov creates *.gcov* file by reading the created *.gcda* files.
4. Finally, it reads the created *.gcov* files and calculates the number of covered branches.

```
If defined, the environment variable `M4PATH' is a colon-separated list
of directories included after any specified by `-I'.

Exit status is 0 for success, 1 for failure, 63 for frozen file version
mismatch, or whatever value was passed to the m4exit macro.

Report bugs to: bug-m4@gnu.org
GNU M4 home page: <https://www.gnu.org/software/m4/>
General help using GNU software: <https://www.gnu.org/gethelp/>
./m4 --version
m4 (GNU M4) 1.4.19
Copyright (C) 2021 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <https://gnu.org/licenses/gpl.html>.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Written by René Seindal.
----------------------------------------------------------------------
----------------Results-----------------------------------------------
----------------------------------------------------------------------
The number of covered branches: 117
The number of total branches: 9471
----------------------------------------------------------------------
```

** Your goal is to increase the number of covered branches!! **

After checking the results, you just submit a text file.

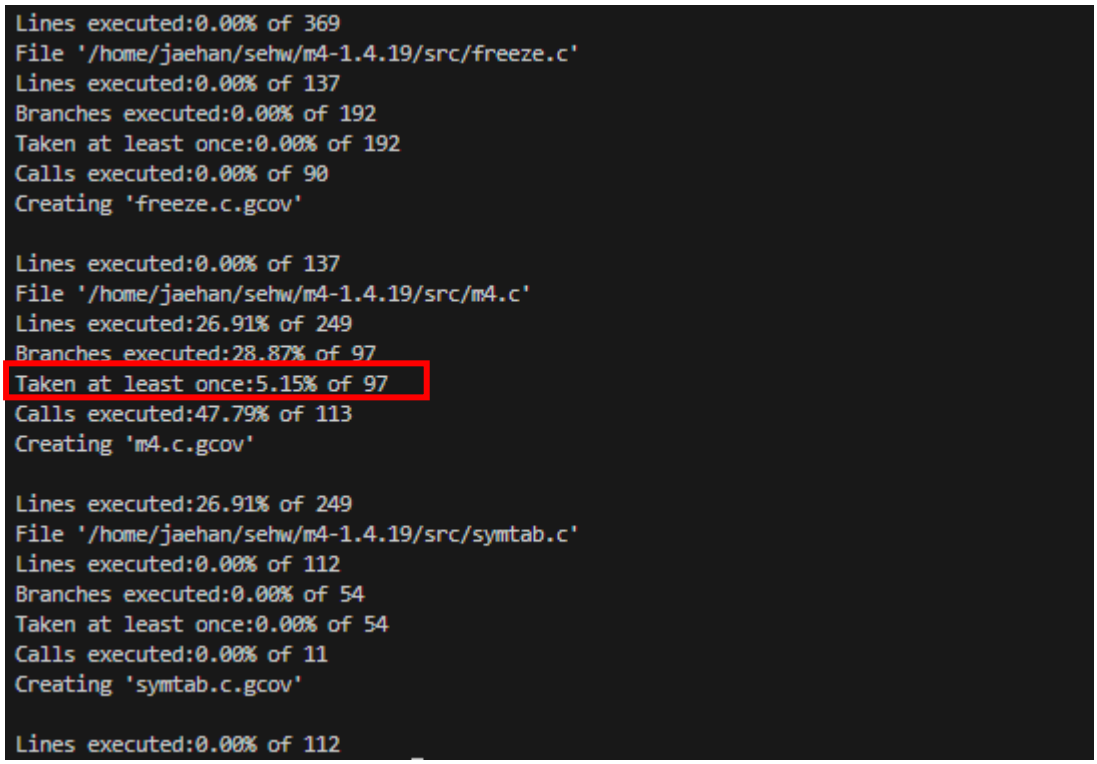
---

4. How to check the coverage of your each testcase

```
$ cd 2025s_hw1/m4-1.4.19/src
$ ./m4 --help
$ find .. -name "*.gcda" -exec gcov -bc {} \;
```

Through this, you can check the number of covered branches for each test-case.

- In this case, `./m4 --help` is the testcase (command).
- That command (testcase) creates the *.gcda* file for each source codes in `2025s_hw1/m4-1.4.19` directory.
- Last command will check the number of covered branches by creating the *.gcov* files from *.gcda* files



** The number of covered branches is calculated as follows **

For each file,

Taken at least once: 5.15% (← covered branch ratio) of 97 (← the number of total branches)

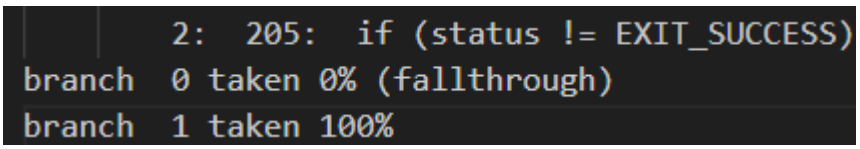
*The number of covered branches = 0.515 * 97 = 50*

→ And we calculate the number of covered branches for all files and **add them together**.

5. How to check the covered branches in each source code by your testcases

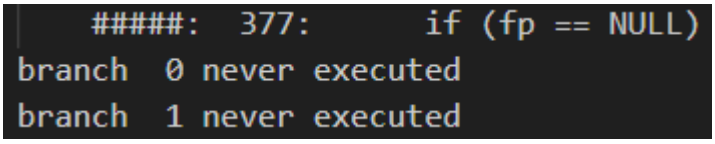
```
$ cd 2025s_hw1/m4-1.4.19/src
$ ./m4 --help
$ find .. -name "*.gcda" -exec gcov -bc {} \;
$ vi m4.c.gcov
```

- Command creates *.gcda* file
- gcov creates *.gcov* file, which you can read by the "vi" or "cat" command
- *.gcov* file shows you which branch is covered
- You can search "branch" keyword in *.gcov* file



In line 205: if (statis != EXIT_SUCCESS) → (*in line 661 in m4.c.gcov*, if statement's branch 0, branch 1)

- branch 0 taken 0% → means this branch is not covered
- branch 1 taken 100% → means this branch is covered



In line 377: if (fp == NULL) → (*in line 377 in m4.c.gcov*, if statement's branch 0, branch 1)

- branch 0 never executed → means this branch is never executed
- branch 1 never executed → means this branch is never executed