HW03: Mutation Testing as a Class

In this homework, you'll be taking the 3 sort programs from Homework 2 and introducing new bugs into them (bugs different than the ones you fixed the first time around). In addition, you'll also develop a test suite for each program that tests the happy path (good cases) but also fails on a test that demonstrates your added bug.

Then, in class, ch student will exchange their buggy programs and test suites with a partner. The objective is to challenge each other's code by introducing mutations (small code changes) and checking whether the provided test suites can effectively detect these changes, thus identifying the injected bugs.

Learning goals

- Gain a better understanding of common programming mistakes and identify issues in code collaboratively.
- Learn to develop a comprehensive test suite for variations of a program, covering both the expected, "happy path" behavior and edge cases that reveal bugs.

Tasks

You will be asked to perform the following series of tasks:

- *Create* your homework-3 repository, with the GitHub classroom invite and clone that repository locally. It uses the hw03 template
- *Copy over* the 3 `sort*.c0` programs **that you've fixed** from HW2 into the new repository
- ***Add 1 bug to each program*** that's different from the ones you fixed up in HW2. We're not looking for compile-time bugs, but logical bugs that can produce incorrect outputs.
- ***Add a test file for each program*** with **at least 1 test** that fails due to the bug that you introduced, and at **least 2 tests** that pass for a happy path. For the happy path tests, this could involve testing specific functions that work, even if the final output is wrong (due to your added bug).
- *Open a Pull Request* (PR) and then **merge it**.
- **Challenge**: Add covering contract code to all functions (except main) and loops (loop invariant) into 1 of the 3 sorting programs.

We expect that you'll be using the C0 VSCode Extension to help to write your C0 programs. This assignment is setup with a dev container, further instructions on how to use it at the beginning of hw02

Generate Your repository and clone it

As is now the norm in this class, accept the GitHub classroom invite at https://classroom.github.com/a/HoMmK5qA, and it will generate a private repository for you, something like `homework-3/<your GitHub username>`. Then clone the repository that you created to a local directory on your machine.

Copy over the 3 `sort*.c0` programs that you've fixed from HW2 into the new repository

Copy the fixed programs over. Update the given favorites.txt as you see fit, as it should still be the input for these programs.

Add 1 bug to each program that's different from the ones you fixed up in HW2

Introduce a non-obvious bug into each program that's not the same as the one you fixed previously. Just for context, again, these sort programs sorted your favorites in alphabetical order like so:

Given a file consisting of:

```
Sit Down for Dinner
Misery Is a Butterfly
Penny Sparkle
Fake Can Be Just as Good
Melody of Certain Damaged Lemons
```

The output should be:

```
Fake Can Be Just as Good
Melody of Certain Damaged Lemons
Misery Is a Butterfly
Penny Sparkle
Sit Down for Dinner
```

Add a test file for each program

Add a test file for each program **with at least 1 test** that fails due to the bug that you introduced, and at **least 2 tests** that pass for a happy path.

For the happy path tests, this could involve testing specific functions that work, even if the final output is wrong (due to your added bug).

For running tests, the README.txt may be helpful.

Note: Remember to rename/comment-out/remote your main function in the `sort*.c0` programs, as you'll have main in your test programs.

Use the following code to get an array of your favorites:

```
bundle_t S = read_lines("favorites.txt");
int length = string_bundle_length(S);
string[] favs = string_bundle_array(S);
```

Open a Pull Request and then merge it into the main branch

For this assignment, you should check out your branch off of the main branch, and push up this branch consisting of commit(s) for the 3 updated sort programs (now with a new bug!) and your test files for each program. Once this is all good to go, please open a Pull Request and then merge it into the main branch.

Challenge: Add contracts to all functions and loops (loop invariant) into 1 of the 3 sorting programs.

Add contracts (which we'll be discussing more in class next week) into one of the sorting programs of your choice. Due to your introduced bug, it's okay for the contract to fail (you can turn on/off contracts via *-d* or *--dyn-check* when using `cc0`.

Resources

1. https://c0.cs.cmu.edu/docs/c0-reference.pdf
2. https://jamiemorgenstern.com/teaching/su-122/lectures/01-contracts.pdf
3. https://visualgo.net/en/sorting

Deadlines and Deliverables

Due Date: Monday, November 11, 2024 at 11:59 pm.

Deliverable: For this assignment, you have one deliverable.

1. You will either upload your GitHub repository URL to Canvas under assignment 3 (embedded via Gradescope as a PDF (with just your andrewID and link it) or DM staff over Slack.

Assignment Review

Because this is a new class, we are asking you to fill out a short survey to help us calibrate the homework. This survey is ungraded, but your input will be very valuable for us in improving the course both for this semester and for future years. Fill out a short survey to help us improve the course!

Grading

The total assignment is worth 100 points, but you can receive an extra 10 points by doing the Challenge question.

Item	Points
Add 1 (new) bug to sort1.c0	15
Add test file for sort1.c0	15
Add 1 (new) bug to sort2.c0	15
Add test file for sort2.c0	15
Add 1 (new) bug to sort3.c0	15
Add test file for sort3.c0	15
Committing to GitHub and opening/merging a Pull Request	10
Challenge: Add contracts to one program	+10
Total	110