HW02: Expectations in Code

In this homework, we'll be reviewing and debugging different programs that all compile, but return unexpected results. These programs do not include contracts or tests. You'll be documenting your process while debugging buggy code.

Learning goals

- Reasoning about a program and/or algorithm without having to jump in and just write code for it
- Walk through the control flow of a program and deduce its errors
- Learn to follow (and document) the scientific debugging process

Tasks

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

- *Create* your own homework-2 repository, with the GitHub classroom invite, https://classroom.github.com/a/ArSRiKrc and clone that repository locally
- *Copy over* your favorites.txt file from HW1 and override the favorites.txt in the HW2 repository (hint, this is a good place to make a commit)
- ***Pick 2 out of the 3*** `sort*.c0` programs that you'll provide writeups/bug reports on and then attempt to fix. They are named `sort1.c0`, `sort2.c0`, `sort3.c0` in the repository. The repository has instructions for how to run them in README.txt, as they all compile, but have incorrect logic. They all specify one thing: take your favorites.txt file from HW1 and sort them in **ascending alphabetical order** (A-Z order). However, they all miss the mark and return different orders for different reasons.

  There's a bug report template available in the repository. You can find it here.
- Open a Pull Request (PR), **but do not merge it yet**. We'll use the PR to give you feedback.
- Challenges:
  - *Provide* an additional bug report/writeup, completing all 3 sorting programs
  - *Answer* the bonus question(s) for each program

We expect that you'll be using the C0 VSCode Extension to help to write your C0 programs.

Note: The README.txt showcases how to compile and run the programs with the `cc0` compiler. For those taking 15-122, you should experience running `coin` as well, which is an *interactive interpreter*. So, there are two ways to run

C0 programs! We can use the `coin` interactive interpreter, which evaluates expressions or executes statements after a program has been loaded, and `cc0`, a compiler which produces an executable binary. We won't cover these differences now, but please feel free to use `coin` if you're comfortable with it.

Generate Your own repository and clone it

Just like last week, there is a repository that has already been set up for HW2. First, accept the GitHub classroom invite at https://classroom.github.com/a/ArSRiKrc. and it will generate a private repository for you, something like `homework-2/<your GitHub username>`. Then clone the repository that you created to a local directory on your machine.

Refer to homework 1 for information on getting setup with GitHub.

Copy over favorites.txt from HW1 into the repository for HW2

Override the contents of `favorites.txt` in the HW2 repository with your favorites list from HW1.

Provide writeups/bug reports on 2 out of the 3 sorting programs

Each of the `sort*.c0` programs you choose from compile and print a sorted output of your `favorites.txt` list; they just sort them with different approaches. The goal is to render printed output in alphabetical order.

For example, 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
```

However, none of the programs in the repository follow the simple spec (ascending alphabetical order [A-Z order]) and, instead, print varying orders. Following the guidance from lecture 2, in a separate file called `bugreport_sort*.txt`, please create a bug report for each sort program you choose (e.g. `bugreport_sort1.txt`, `bugreport_sort2.txt`, `bugreport_sort3.txt`) and include:

- **the name of the sorting function**

- **the defect observed**, demonstrating the contents of your favorites.txt file and the expected and incorrect outputs
- **the hypothesis about what is going wrong** in the program(s)
- **Experiments that can confirm or reject the hypothesis**. These might include:
  - adding print statements to the code at certain lines
  - making changes to the code and running with different variations of the input file
  - etc.
- Upon experimenting, **each hypothesis should be confirmed or rejected**, and then a process of iteration should occur to either develop a new hypothesis or run a new experiment.

Note: There are hints (**marked with a Q**) that reference what/where the issues may be. There's also a template file setup to fill out this information here.

Fix the bugs in the programs!

Now that you have bug reports, fix the bugs and update the files (for the 2 you've chosen)!

Open a pull request to the main branch

Once you've written up a bug report for each sort, e.g. `bugreport_sort1.txt`, and have committed your changes and fixes, you should push the branch to the GitHub repository and open a pull request to the main branch. GitHub's UI will even demonstrate this to you. Leave this as an open Pull Request, as we'll review it and possibly ask some additional questions.

Challenge 1: Provide an additional writeup/bug report and fix the leftover program

Provide another bug report and fix the code for the other program leftover, completing all 3 sorting programs.

Challenge 2: Answer the bonus question(s) for each program

The bonus question for each program is always the same:

> What algorithm do you think is being implemented in this program?

Resources

1. https://c0.cs.cmu.edu/docs/c0-reference.pdf
2. https://visualgo.net/en/sorting

Deadlines and Deliverables

Due Date: Wednesday, November 2, 2023 at 11:59 pm.

Note: You can get an **extra 5 points** by committing the write-up/bug report for one of these programs by **Monday, October 30, 2023** at 11:59pm to your repository.

You may use up to two (2) late days, out of four total. To use a late day, simply message us on Slack to let us know that you will be using a late day.

Deliverable: For this assignment, you have one deliverable.

1. You will either upload your GitHub repository URL to Canvas under assignment 2 (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 contents are ungraded, but will give a participation point for filling it out, and 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 15 points by doing the Challenge questions, and an extra 5 points by committing a write-up for one of the programs by October 30, 2023.

Item	Points
Fixing Bug(s) #1	5
Writeup/Bug report #1	40
Fixing Bug(s) #2	5
Writeup/Bug report #2	40
Committing to Github and opening a Pull Request	10
Total	**100**
Early Commit / Submission (DUE Oct 30, 11:59pm)	+5
Challenge #1: Writeup/Bug Report & Fixing Bug #3	+10
Challenge #2: Bonus Question(s)	+5
Optional Bonus	**20**