HW01 - Version Control with Git

In this homework, you will be practicing using version control (git and GitHub). We will be using this throughout the course extensively.

Learning goals

- Understand the basic concepts of Version Control Systems (VCS)
- Create a GitHub Account
- Perform basic VCS actions
- Understand the difference between git and GitHub
- Practice VCS commands necessary for turning in homework

Tasks

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

- *Create* a GitHub account (if needed)
- *Create* your own *homework-1* repository, with the GitHub classroom invite, and clone that repository locally
- *Create* a branch locally
- *Make changes* to the student.txt and favorites.txt files
- *Update* the `print_favs.c0` C0 program to print the index of each favorite in `favorites.txt` next to the item
- *Push* that branch to GitHub
- *Open* a Pull Request to the main branch and merge it
- **Challenge**: *Manage* a file conflict

All of these tasks can be done via the command line or through VSCode, whichever you're more comfortable with.

To install `git`, please follow this lovely guide! We highly recommend that you do not use *Github Desktop*!

To install C0, please follow along with our README and associated links.

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

Create a GitHub account

You should create a free GitHub account if you do not already have one. If you are creating a new GitHub account, you might want to consider using your andrewID, as that will make things easier for us. But you may use any account name you want (within reason, ask the instructors if you have questions).

If you have an existing GitHub account, you may use that.

Generate your own repository and clone it

There is a repository that has already been set up for HW1. First, accept the GitHub classroom invite at https://classroom.github.com/a/Qal8n-kz and it will generate a private repository for you, something like `homework-1/<your GitHub username>`. Then clone the repository that you created to a local directory on your machine.

Create a branch locally

You should create a new branch of the repository, and you should name the branch with your andrewID as the name.

Make changes to the `student.txt` and `favorites.txt` files and commit them locally

In the branch that you've created, you should make changes to the `student.txt` file, adding your name and andrewID (replacing the one there). You should also update/replace the favorites.txt with at least 5 books, films, artists, albums, or related cultural items that you enjoy (it's a list of your favorites after all!).

You should then commit these changes to your local branch.

Update the `print_favs.c0` C0 program

Inside the repository is a file named `print_favs.c0`, which reads your `favorites.txt` file into an array and then loops through and prints each item. The repository has instructions for how to run this program in README.txt.

For this assignment, we would like you to update the code to print a number next to each favorite item that is printed. For example, given the example file in the repo, running the program should give us this:

```
wings
the ascent
farewell
you and me
heat
--------------------
...
```

Your updated output should display the index of the item next to the name of item, like so:

```
(0) wings
(1) the ascent
(2) farewell
(3) you and me
(4) heat
```

```
------------------
```
...

That's it! You should then commit this change to the same local branch as another commit.

Push the branch to GitHub

Once you have committed your changes, you should push the changes on the branch to the GitHub repository. Please ensure you use a descriptive commit message! We should be able to decipher what you did from this message.

Open a Pull Request to the main branch

Now that the commit is visible on the GitHub site, you should open a Pull Request from your branch to the main branch. Again, don't forget to have a descriptive message for the Pull Request.

Challenge: Manage merge conflicts

Merge conflicts can be very annoying, and will be a cause of headaches for you. We want you to try and follow these steps to (artificially) create a merge conflict so that you can merge the conflict. Here are the steps to follow:

- In the branch , create a text file called `student_facts.txt`
- In this file, write what year and college you are in
- Commit this change and include a message about what you just did
- Create a new branch called <new_andrewID> and switch to that branch
- Open up `student_facts.txt` and delete the previous line you wrote and now only include a sentence about a fun fact about yourself
- Commit this change and include a message about what you just did
- Switch back to the branch
- Open up `student_facts.txt` and append to this file a sentence about what programming languages and/or coding classes you have taken
- Commit this change and include a message about what you just did
- While in , merge this branch with <new_andrewID>
- You will see that there are conflicts
- To resolve these conflicts, open the file `student_facts.txt` and edit the file such that there are three different lines (and in this order):
  - A sentence about your year and major
  - A sentence about what programming languages and/or coding classes you have taken
  - A sentence with a fun fact about yourself
- Commit this change and include a message about what you just did
- While in , create a file with all of the Terminal commands you used so far

Resources

- https://guides.github.com/
- https://guides.github.com/activities/hello-world/
- https://gist.github.com/davfre/8313299
- https://docs.github.com/en/repositories/creating-and-managing-repositories/cloning-a-repository
- https://code.visualstudio.com/docs/sourcecontrol/intro-to-git
- https://learn.microsoft.com/en-us/visualstudio/version-control/git-with-visual-studio

Tips and Suggestions

Writing a good commit message is hard! Don't feel bad if it takes a bit of thinking, or if you are not happy with your first version.

One pro tip with git: it can often be the case that your local directory can get messed up. Feel free to delete your local directly and start over. This is a common technique when working with git.

Deadlines and Deliverables

Due Date: October 25, 2023 at 11:59 pm.

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 1 (embedded via GradeScope) as a PDF (with just your andrewID and link in 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
Create a GitHub account	20
Make/update and commit changes (2 commits, 3 files)	40

Item	Points
Correctly open Pull Request	40
Challenge: Manage a Merge Conflict	10
Total	**110**