1 Homework 1: Warm-Up

Due: Sunday 9/17 11:59pm

Late submissions accepted until Tuesday 9/19 11:59pm, with a 15% late penalty. If you have already created a `hw1` branch and want to try again to do better, you can create a new branch named `hw1a`, or `hw1b`, or `hw1c`, and so on. I will grade the alphabetically-latest such branch in your repository. You must submit a Regrade request on Gradescope for your late submission to be graded.

Note: *The rules on collaboration are slightly relaxed for this assignment. You may get help by allowing other people to look at your work. You still* ***may not*** *share written code (including literal commands to run), and you* ***may not*** *allow others to use your mouse or keyboard.*

1. Log in to gitlab.com.

   Perform the following tasks using the gitlab.com website:

   a. Go to the "course repo" at https://gitlab.com/rmculpepper/cs410-f23.

      You should have received an email inviting you to the repository, with one of the following subjects:

      - "Access to the Ryan Culpepper / cs410-f23 project was granted" —Your GitLab account was associated with your `@umb.edu` email, and you were automatically granted access.
      - "Ryan Culpepper invited you to join GitLab" – The email has a "Join Now" button/link. Click it for more instructions.

   b. Fork the course repo. The rest of these instructions will refer to that fork that you created as "your repo" or "your fork".

      When creating the fork, you must choose a `Project URL`. You must use your user name as the "namespace" part, and you must leave the "Project slug" as `cs410-f23`.

   c. Make sure that your repo is *private* by going to `Settings > General > Visibility, project features, permissions` and checking the `Project visibility` setting.

   d. Add your instructor (user `rmculpepper`) to your repo with the role of `Developer`. Do this by going to `Manage > Members`, then click `Invite Members`. Do not add anyone else to your repo.

2. Make a local working copy (aka "clone") of your repo. (This is sometimes called "checking out" the repository, but Git mainly uses "checkout" to refer to switching the branch you are actively working on.)

   Perform the following tasks on your working copy:

   a. Create a branch called `hw1` based on the `start-hw1` branch. Check out `hw1`.

b. Edit the file `hw1/info.json` according to the directions below. Commit your changes.

c. Merge the course repo's `fix-hw1` branch (probably visible to you as `origin/fix-hw1`) into your `hw1` branch. Fix the merge conflicts. (Do not `rebase`, if you know what that is. You must create a merge commit.)

d. Use IntelliJ to build and run the Scrambler project according to the directions below.

e. Push your `hw1` branch to your repository on GitLab.

3. Check your work.

1.1 Edit `hw1/info.json`

Edit the JSON object in `hw1/info.json`.

First, correct the values of the `"course ID"` and `"course name"` keys. Do not change the names of the keys.

Then add the following information to the JSON object using the given keys exactly. Add your new entries after the existing entries.

- `"name"` — (String) Your name.
- `"major"` — (String or `null`) Your declared major.
- `"PLs"` — (Array of String) Programming languages that you have some experience with.
- `"400-level prerequisite"` – (String or `null`) The 400-level course that you have taken to satisfy the CS410 prerequisites.
- `"favorite algorithm"` — (String) The name of your favorite algorithm.
- `"favorite data structure"` — (String) The name of your favorite data structure
- `"CS interests"` — (Array of String) A list of areas within Computer Science that interest you.
- `"non-CS interests"` — (Array of String) A list of things outside of Computer Science that interest you.

Make sure that `hw1/info.json` is valid JSON and that each key has the requested type of value.

1.2 Build and run the Scrambler

1. Use IntelliJ to create a new project called Scrambler, and copy the Java file at `hw1/Scrambler.java` into the project. *Warning:* Make sure that the new `Scrambler.java` is in the right directory, and make sure that IntelliJ does not "helpfully" delete the package declaration!

   For the build system, select either IntelliJ or Maven. Do not select Gradle.

2. Build the project.

3. Run the Scrambler on your `@umb.edu` email address. Commit the result as a new file called `hw1/scrambled-email.txt`.

4. Use the JShell Console to evaluate `scramble("Software Engineering", 410)` *without editing the Java source file!* Commit the result to a new file called `hw1/scrambled-cs410.txt`.

   Warning: Configuring JShell Console in IntelliJ is somewhat finicky and unintuitive. Follow the JShell Console tool guide carefully. In particular:

   - In the Libraries page, you must add the path that immediately contains the `cs410` directory of compiled `.class` files. If you selected the IntelliJ build system when creating the project, the path should look like `.../hw1/Scrambler/out/production/Scrambler`. If you select the Maven build system, the path should look like `.../hw1/Scrambler/target/classes`.

   - In the Modules page, make sure the `Scrambler` library is listed in the Dependencies tab. (It does not appear to matter whether the checkbox is checked.)

5. Check in your project directory.

1.3 Check your work

Added 9/16, around 8pm.

Visit your fork on gitlab.com.

Go to `Manage > Members`. You should see two users listed: your username listed as “Owner” and mine (`rmculpepper`) listed as “Developer”.

Go back to your fork’s main page. The page should have a drop-down box listing your repo’s branches; it probably has `main` selected by default. (The drop-down box should be right above the box saying “Forked from ...”.)

Click the box. There should be a branch named `hw1`. Select it, then click on the `hw1` directory and verify that it contains all of the files you were expected to check in.

Your fork may have other branches, too, but `hw1` is the one that matters.

Note: You should *not* create a “merge request” (also known as a “pull request”) with your changes. If you have created a merge request, you should close it.