Northeastern University
CS5500 – Managing Software Development
Fall 2018, {Jump, Katz, and Weintraub}

Using git

The purpose of this exercise is to introduce you to git and github. It assumes you do not know git. By the end of this homework, you will have cloned a repository. You will also have made changes to files, committed them, worked on branches, merged these, and then pushed the results back to origin repository.

If you have worked with git and github, please pay attention to the requirements of each step, as you may have used git differently (for example, in a co-op, they may have a different policy for managing branches. And, if you don't know what a branch is, don't worry, you will soon enough). Not following the directions will cause you to lose points.

At different points throughout this assignment, you will be asked to enter specific text strings. Please pay careful attention to these instructions as failure to follow the instructions will very likely mean your work will be marked as wrong.

You will also be asked questions throughout the steps (marked **QUESTION:** in a step's text). Please put your answers in a file called `homework1` that must be in the `Homework 1` directory. Please label each answer with a bold **STEP** X where X is the step number in which the question was asked. Often, the best answer will simply be to cut and paste the information from the terminal. If you do this, it would help if you include the git command along with the results. `homework1` may be a txt, doc, or pdf.

This assignment will be graded on whether you are able to complete the steps, the answers to the questions, and the quality of your software.

Prep Step One: Get git

If your system does not have git installed, install git from `https://git-scm/downloads`. We suggest you accept the defaults.

Prep Step Two: Open a shell/terminal window

1. If you have a mac or use linux, open a terminal shell.
2. If you use Windows, open `git bash`. Cortana is often helpful here.

All work must be done through the terminal. You may not use graphical clients for this assignment unless the step allows it explicitly. This includes your IDE (e.g. Intellij or Eclipse). Knowing how to use git from the command line will cement an understanding of how to use git (and github).

Assignment Part One

Do the following tasks in order:

1. Configure git with your name, email, and favorite editor. Verify git knows your name, email, and editor by having git list your configuration.

   QUESTION: What are your git config's?

2. You will now be creating and modifying files in a project. The project is already set up for you on github.ccs.neu.edu. The url is `https://github.ccs.neu.edu/CS5500/student-XXX-F18` where XXX is assigned to you. Clone this repository to your machine using git's `clone` command. Show git cloned the repository by listing the contents of the parent directory and then creating a listing that includes the `.git` directory.

   QUESTION: What files or directories are in the `.git` directory?

3. Edit the repositorys README.md to describe your repository. Remember, the README.md is for people who are looking for a repo, not the author. Be creative and try something beyond: *this is the repository for Your-Name-Goes-Here*. You will be using this repository for your homework submissions. See `https://guides.github.com/features/mastering-markdown/` for formatting commands.

   QUESTION: Using markdown, how would one express a link to the course web page (`https://course.ccs.neu.edu/cs5500`)?

4. Decide which files you do not want git to keep track of or allow to be committed to source control. Set up your `.gitignore` appropriately. You may want to use the templates others have contributed as a guide (see `https://github.com/github/gitignore` for a raft of suggestions across many technologies). You do not need to use these files in part or in whole; they are suggestions. Commit these rules in the project. Insure the text **setting up gitignore** appears in your commit message.

   QUESTION: What rules are in your `.gitignore`?

5. Create a branch called `feature-1`. Switch to that branch and move into the `Homework 1` directory. All your work will be done in this directory and on this branch.

6. Using an IDE of your choice, write a java program that takes as input two numbers (read from the console) and prints out all the numbers that sit between these two numbers,not including the two inputs in the range. You should follow *test-driven design*, meaning you should create test spec's / junit test cases in junit after you define interfaces and ***before*** you code the methods. Your code should be properly commented using javadoc. Your coding project should have a `src` and `test` directories.

   Once the code works, add the project files to `git` and commit them. Please insure the text *satisfying requirement one* is in the commit message. Notice we are using the commit message to cue readers to the driver of the change, not describing the change itself (they can look at the code for that). And remember, you should not commit intermediate or local files.

   QUESTION: What are all the commits you've made on the `feature-1` branch? What files and sub-directories are listed in your `Homework 1` directory on the `feature-1` branch. What files and sub-directories are listed in your `Homework 1` directory on the `master` branch. Insure it is easy to tell which listing goes with which branch.

7. Now merge `feature-1` into `master`.

8. Create a new branch called `feature-2` off master.

9. Create a new branch called `feature-3` off master.

10. Switch to the `feature-2` branch. Change the method that prints the numbers in the range so it now prints only the odd numbers in the range. *Do not add a method. You must change the existing method.*[1] Adjust the tests to reflect this new requirement. Commit the changes on `feature-2`. Please insure the text *satisfying requirement change two* is in the commit message.

    QUESTION: What are all the commits you've made on the `feature-2` branch?

11. Switch to the `feature-3` branch. Change the method that prints the numbers in the range so it now prints only the even numbers in the range. *Do not add a method. You must change the existing method like you just did in step 10.* Adjust the tests to reflect this new requirement. Commit the changes on `feature-3`. Please insure the text *satisfying requirement change three* is in the commit message.

12. Merge `feature-2` into `master`. Resolve any merge conflicts so `master` reflects the functionality you created in `feature-2`.

13. Merge `feature-3` into `master`. Resolve any merge conflicts so `master` reflects the functionality you created in `feature-3`.

    QUESTION: What are the differences between master and the other three branches? Hint: use `git diff`.

14. Oh no! After merging `feature-3` into master (in step 13), you realize that the client changed the requirements of the project. In fact, the only feature that they want is `feature-2`. Create a new branch off `master` called `revert-feature-3` and "`git revert`" your changes so the branch only reflects the changes made by `feature-2`. Merge this commit into `master`.

    QUESTION: What are the differences now between master and the other three branches? Hint: use `git diff`.

15. Push *all five branches (master, feature-1, feature-2, feature-3, and revert-feature-3)* to `origin`. You can verify the results by going to `https://github.ccs.neu.edu/CS5500/student-XXX-F18` and looking at the `Homework 1` directory. You should be able to see all the branches and the commit log.

16. Insure the answers to all the questions are in `homework1` and push it to `origin`. `homework1` MUST be in your `Homework 1` directory and MUST be found on `master`.

[1] If you are wondering why, we want to induce a merge conflict.