Software Engineering Exercise 2:

Collaborative Development With GIT

DUE: Monday, January 23, 2016 @ 5pm.

Fun w/ GIT

The general exercise goal here is familiarity with GIT. Just as with any other new technology you are exposed to; the more you use it, the easier it becomes to use. A secondary goal is to help you develop your problem solving skills through research on the Internet and use of tutorials and manuals.

Read this entire document before you begin as it will help prevent many issues that will inevitably come up

This assignment has 2 components that you must accomplish

1. In your github account create a repository and make sure you know how to pull, push, and make changes.
2. You will be editing a repository that includes all of the students in this class and you will all be working together to create a story.

First part

- In your account create a new repo
- In this repo, you will need to clone it so that you can edit it (in whatever software you prefer, though command line will work if you chose to use that)
- Create a branch
- In this branch create a file and write something in it
- Now merge this branch with the master branch in your repository.
- Take a screenshot of the commit (repo page -> commits -> your commit) which you will submit with the files from the second part

Now onto the repo you will all be part of….

Second part

Remember, there are ~120 students that will be contributing to this repository and there should be ~120 branches pushed up to me when this is all said and done.

STEP 1

- View the repository website. Determine the manner in which you should clone the repository and any necessary step.
- **https://github.com/Hackers-To-Engineers/GitMagic**
- **WARNING: Downloading a ZIP or TGZ ? Nope… this means you're doing it wrong!**

STEP 2

- Test that what you pulled down actually compiles... or do you get to fix someone else's errors?

```
sgoggins@ceberus:~/GitMagic$ make                    // RUN “make” TO BUILD PROGRAM, “all” targe
gcc -c story.c
gcc -c sentences/bpbkt7.c
gcc   story.o bpbkt7.o   -o story
scottgs@ceberus:~/GitMagic$ make test                /// RUN “make test” to run the test target
```

Once upon a time

The end.

```
scottgs@ceberus:~/GitMagic$ make clean               // CLEAN UP before you get started
```

Make sure you read the story so you can make it better!!!!

Consult the great and powerful Google. **Branch** the *master* into a new branch that is the name of your University Pawprint

Review the following files that are part of the code:

1. source code in story.c
2. Makefile
3. sentences/bpbkt7.c
4. sentences/_HEADERS.h

Add at least two non-consecutive sentences to the story by completing the following steps:

1. create a file in the sentences folder named after your pawprint, it should include at least two different function to calls that each have a single **printf** of a sentence.

For example, see the **sentences/bpbkt7.c**

2. Edit the sentences/_HEADERS.h file as expected
3. Edit the story.c file call your sentence in the appropriate spot… after the previous person's sentence and before the end.
4. Modify the Makefile to:
    1. Compile your source file into an object file
    2. Link your object file into the program
5. Build, run, confirm

STEP 3

When you first begin to register activity into the repository, it may ask you who you are

*** Please tell me who you are.

Run

git config --global user.email "you@example.com"

git config --global user.name "Your Name"

to set your account's default identity.

Omit --global to set the identity only in this repository.

fatal: unable to auto-detect email address (got 'sgoggins@goggins.(none)')

Apply common sense to interpret this message

Example, I did the following ... you should substitute you:

git config --global user.email "gogginss@missouri.edu"

git config --global user.name "Sean Goggins"

Consult the great and powerful Google and **pull down an update of master**

1. Result conflicts? Then fix them
2. Does is build OK?
3. Merge your branch into master
4. Does is build OK?
5. YES: Commit and Push
6. NO: Resolve issues, Commit and Push *master*
7. Switch to your branch and Push

STEP 4

- Do your happy dance

STEP 5

Document your completion of the exercise

- Switch to your branch and produce a GIT log using:

  git log --stat > branch_*pawprint*.log

- Switch to the master branch and produce another log using:

  git log --stat > master_*pawprint*.log

- Zip the two files along with the screenshot from the first part together

- Submit to canvas before due date

==========
Example GIT WORKFLOW

- git status
- (Are you on your branch?)
- Make changes
- git add .
    - This is shorthand for git add – - all. Adds all the files that aren't staged. It isn't always the right way to go. But, often it is.
- git commit -m "Type your detailed commit message here"
- git push
- (Switch back to master: git checkout master)
- git merge your_branch_name_here
- git push
- Notes:
    - When you hop on for the first time every day to work, checkout the master, pull down any changes with 'git pull'. Switch back to your branch and type git merge master. Proceed with step 1!
    - Address any conflicts. You've just merged the master into your branch. Then start with step 1