The makefile and git
Lab H

TA teaching instructions can be found at the end of this document.

The labs, for this course, are designed to be completed on your own at home or in the 3rd floor Trottier labs. These labs are not graded. You do not hand in these labs. If you prefer to work on a lab with your TA tutorial group, then check the schedule for your TA's tutorial session. You will find this schedule in our MyCourses page under Content/Course Information/TA Information. Since the university has limited lab space, your TA might ask you to bring your laptop and work in a classroom instead of a lab.

This lab is about programming with struct / union and malloc / calloc.

Some labs will have a question zero. These questions will not be covered by the TA during the tutorial. It is extra content meant for you to do on your own.

QUESTION ZERO: Optional problem

If you like, review these online resources:

1. The makefile: https://opensource.com/article/18/8/what-how-makefile
2. The git repo: https://www.freecodecamp.org/news/what-is-git-and-how-to-use-it-c341b049ae61/

QUESTION ONE: The makefile

Copy the main.c and bank.c files from Lab G Question 1, then do the following:

1. Create a file called bank.h and move the BANK_ACCOUNT type definition into it.
2. Create a makefile for this project: main.c, bank.c and bank.h
3. Use **make** to compile the project
4. Run the program to make sure it still works

NOTE: See the sample makefile in the TA Teaching Instructions section on the last page of this lab for inspiration.

QUESTION TWO: The git repo

Reusing Question One, do the following:

1. Create a git repo in the directory where the makefile exists: `git init`
2. "Stage your commit" by adding the files you want to put into the repo: `git add`
3. Do the commit: `git commit -m "some initial message"`
4. Now edit one of your files, re-stage that file, and commit. When editing the file you could do anything, like adding a comment.
5. Do this again with another file.

6. Now do: `git log`, notice how the SHA digits and your messages look like.
   Where your messages useful?
   How could you have made them better?
7. Create a branch, and edit a file in that branch, then stage and commit within that branch.
8. Switch to the master branch and vi the file you previously edited in the branch to see that the changes are NOT there. Then merge the branch to the master and vi the file again to see that the changes you made ARE now there.
9. Do a git log to see what it looks like now with the branch.
10. Select a SHA from the log randomly, revert your current files to that time:
    `git checkout SHA`. Do not select the most recent SHA.
11. Vi your current files to make sure that they have been reverted.
12. Now revert again to the tip files (go back to the most recent files in the repo). Using vi verify that the files have been reverted correctly (they should have the most recent changes).

You have completed your lab.

TA Teaching Instructions

At the beginning of the tutorial, ask the students if they would like to do the lab on their own or if they want you to lead them through the lab. If they want to do the lab on their own, then walk around answering their question. If they want you to lead the lab, then project your screen and go through the lab step by step giving the class time to catch up with you. You may want to review some material before you start the lab. Below are some suggestions:

- Review the basic structure of a makefile, for example:

  ```
  prog: f1.o f2.o
     gcc -o prog f1.o f2.o
  f1.0: f1.c f1.h
     gcc -c f1.c
  f2.o: f2.c
     gcc -c f2.c
  ```

  What does the above mean? How should I use it? Why is it important?

- Review the concept of a repository
- Review the basic git commands: add, commit, diff, log, branching, checkout, reverting a change