CS 4410: Operating Systems
Homework 0

Instructions for Homework 0:

- This is a compulsory homework for all students in CS 4410.
- The homework is to be done individually by each student.
- This homework has two deadlines:
  - All steps in Part I are to be completed by 08/30/2016 at 11:59AM.
  - All questions in Part II are to be completed by 09/01/2016 at 11:59AM.
- Expected time to complete this homework is less than 1 hour.
- This homework is NOT a part of "k out of n" policy.
- This homework corresponds to 1% of your grade.

1 Part I: Preparing for Everything That Comes Next

1. Login to the Cornell GitHub using your NetID and password:

   https://github.coecis.cornell.edu

2. Fill out the following Google form to provide us some necessary information:

   https://goo.gl/forms/hKlv5qX5DSw4tqSg1

3. Wait for your personal repository to be created by the course staff. You will receive an email notification once it is available (please wait for one business day). Once you receive the email, view your personal repository by visiting (replace <NetID> with your NetID):

   https://github.coecis.cornell.edu/cs4410/<NetID>

2 Part II: Preparing for Projects, Homeworks and Submissions

2.1 Preparing to Execute Projects

1. We will use a version control system called **Git** for all project releases and solution submissions. Git is a cloud-hosted service and is extensively used in industry for software development and code management, and in general, is a good tool to know about. The files and instructions for installing Git are here:

   https://git-scm.com/downloads

2. We typically want to build software that can be run across various machines without installing a million libraries on every machine. In this course, this is particularly important so as to avoid questions like: "What do you mean the code does not work? It runs fine on *my* machine.".

   To achieve the above, we will use a tool for configuring virtual deployment called **Vagrant**. Using Vagrant, your code will run inside a "virtual machine" and will (a) avoid the pain of installing all the required libraries; (b) allow us to grade your solutions fairly; and (c) avoid the above question! The files and instructions for installing Vagrant are here:

   https://www.vagrantup.com/downloads.html

3. As mentioned earlier, Git stores your project files on a remote server. The next task is to create a *local working repository* of your Git repository, and to learn how to synchronize your local repository with the one stored on the remote server. Let us start by creating a directory where you will work on all your projects for this course. Let us call the directory `cs4410-projects`.

   Open up a Terminal[1]. Within the terminal, navigate to the `cs4410-projects` directory you created above. We think you should be able to figure out how to do this :-) Once you are in this directory, type and execute the following command (as earlier, replace <NetID> with your NetID):

   ```
   git clone https://github.coecis.cornell.edu/cs4410/<NetID>.git
   ```

   This "`git clone`" command will clone the remote Git repository on your local machine. You will see that you have a new folder, named <NetID> in the `cs4410-projects` directory.

[1] if you are using OS X or Linux; on Windows, it is called Command Prompt. We will just refer to it as Terminal.

4. Navigate into your local working repository using the following command in the terminal:

   ```
   cd <NetID>
   ```

   By now, you must have already figured out what "`cd`" command does. You must already know what to do with `<NetID>`.

5. The next task is to set up the virtual machine. The following command automatically sets up a virtual machine using the Vagrant tool. Please note that this step may take some time, so wait patiently after executing the following command:

   ```
   vagrant up
   ```

6. The final task for this exercise is to access your virtual working environment for the projects. The following command will initiate a secure shell session to the virtual machine. Basically, once you execute the following command, you will be inside the virtual machine and all subsequent commands will be executed on behalf of the virtual machine.

   ```
   vagrant ssh
   ```

2.2 Learning to Submit Projects

1. When a new project is released, you need to first *pull* the new project files from the remote Git repository to your local working repository. This is done using the following command:

   ```
   git pull
   ```

   Note that this is the first time you are executing the above command since you created a local copy of the remote Git repository. Hence, there may be no new updates. You will see that currently there is a folder `10-P0` in your local repository.

2. Navigate to the project folder `cd 10-P0` (by now, you should know how).

3. Follow instructions in the project folder carefully to complete the project.

4. Once you have completed the project, you need to **commit AND push** your code from your local repository to remote Git repository so that we can get access to your solution. To do this, execute the following commands:

   ```
   # Track the changes you have made to your code
   git add <NetID>*

   # Take a snapshot of your current changes
   git commit -m "<a quick message about what this commit changes>"

   # Upload your snapshot (i.e. current state of all source code)
   git push origin master
   ```

5. Verify your submission contains all of your code by browsing your code at https://github.coecis.cornell.edu/cs4410/<NetID>.

6. Note that if you do not execute the above commands carefully, we will NOT have access to your solutions and we will not be able to grade your project.

2.3 Learning to Close and Restart Vagrant

1. Once you have submit your current project solution, you may want to rest for a few minutes (:-)) before starting on your next project. During this time, it is important to not let Vagrant run (since it does use resources that may slow down your computer). You may close Vagrant using the following command:

   ```
   vagrant suspend
   ```

2. Once you have had some rest, you can restart Vagrant using the following command:

   ```
   vagrant up
   ```

2.4 Learning to Read Policies Carefully

1. Carefully read all the course policies regarding communication, academic integrity, late submissions, examinations, etc. at

   http://www.cs.cornell.edu/courses/cs4410/2016fa/policies.html

2. Create a file (either text or PDF, no other formats please). This file should contain one of the following two sentences (replace <..> with the appropriate content):

   (a) My name is <Your Name> and my NetID is <NetID>. I have read all the course policies carefully. I agree with all the policies as stated and will not complain about these policies ever ever again during the fall 2016 term.

   (b) My name is <Your Name> and my NetID is <NetID>. I have read all the course policies carefully. I would like to request you to consider changing the policy regarding <policy_that_you_are_less_happpy_with> because <reason_you_are_less_happpy>; I propose to change the policy to <policy_that_will_make_you_happpier>. I understand you may not change the policy, and in that case, I will keep the right to complain about this policy during the remainder of the semester. I agree with all the other policies as stated and will not complain about these policies ever ever again during the fall 2016 term.

3. Name the above file <NetID>-policy.txt OR <NetID>-policy.pdf based on file type.

2.5 Learning to Submit Homeworks

1. When you filled out the Google form in the first part, you must have received two emails — one about a Git repository, and the other about a Gradescope account. We will use the latter service for all homework submissions. Use this email to login to Gradescope on this website:

   https://www.gradescope.com

2. Upload the file you created above in **HW0 directory on Gradescope**.