Homework 1

File Processing (Term I/2018–19)

built on 2018/09/24 at 16:19:18 **_due:_** *Tue, Oct 2 @ 11:59pm*

Be sure to read this problem set thoroughly, especially the sections related to collaboration and the hand-in procedure.

Collaboration

We interpret collaboration very liberally. You may work with other students. However, each student ***must*** write up and hand in his or her assignment separately. Let us repeat: You need to write your own code. You must not look at or copy someone else's code. You need to write up answers to written problems individually. The fact that you can recreate the solution from memory will be taken as proof that you actually understood it, and you may actually be interviewed about your answers.

Be sure to indicate who you have worked with (refer to the hand-in instructions).

Hand-in Instructions

To submit this assignment, please follow the steps below:

1. Make sure your scripts run and work correctly on Hamachi. We will run the grading script on the server.
2. Zip up all the scripts and name it `a1.zip`

   ```
   > zip a1.zip quota.sh extrm.sh flatten.sh invite.sh happy_countries.sh forkit.txt
   ```

3. Find out the MD5 hash of your zip file. You will need to submit this code on Canvas. We use it for keeping track of your submission time. You may resubmit your work but the MD5 hash has to match.

   ```
   > md5sum a1.zip
   ```

4. Copy the zip file to the directory `/subm/u5712345` where `u5712345` is your student ID.

   ```
   > cp a1.zip /subm/u5712345
   ```

5. Log on to Canvas, go to assignment 1 submission page and enter the MD5 hash.

Task 1: Disk Quota (10 points)

Assume that you were a system admin. Your team was assigned with gathering information of disk space in use. In particular, you are charged with writing a shell script called `quota.sh` to determine how much a particular directory has consumed its disk quota. To make this information easy to understand, we classify a directory as follows:

- **Low** if the directory uses less than 1 MB disk space
- **Medium** if the directory uses less than 10 MB disk space
- **High** if the directory uses equal to or more than 10 MB disk space

Expected output Given a directory, the script should output the classification of the directory's size as `Low`, `Medium`, or `High`.

```
> ./quota.sh /usr/bin
High
> ./quota.sh /tmp
Low
```

Task 2: File Extension Remover (10 points)

Given a specific extension and a directory, write a shell script called `extrm.sh` to remove the extension from the name of any file with that specific extension. For example, assume that the extension `jpg` is given as input to the script along with a path to a directory, and the given folder contains two files that have that extension, named `image1.jpg` and `image2.jpg`. The script in `extrm.sh` should rename these two files these two files to `image1` and `image2`, respectively.

```
> ls images
a.jpg b.jpg c.txt
> ./extrm.sh jpg ./images
> ls images
a b c.txt
```

Task 3: Unzip and Flatten (10 points)

Aj. Sunsern has had headache every term from students' submissions that do not conform to the assignments' file requirements, especially the unwanted directories in the submitted compressed files. Therefore, you are assigned to help Aj. Sunsern out by writing a bash script called `flatten.sh` that decompresses a given zipped file and extracts all files to a specific location. If the destination does not exist, it has to be created. In case of files having the same filename, you are required to keep only one of them.

For example, `handin.zip` contains the following files and directories:

```
handin.zip
 |- a1
 |  |- p1.py
 |  |- p2.py
 |  |- README.txt
 |- p2.py
 |- p3.py
 |- README
```

Here is the expected output of your script.

```
> ls iccs207
handin.zip
> ./flatten.sh iccs207/handin.zip iccs207/submission
> ls iccs207
handin.zip submission
> ls iccs207/submission
p1.py p2.py p3.py README README.txt
```

Task 4: CSV Extractor (10 points)

An organization is organizing an event but, unfortunately, some activities have age restriction of 18 or over. Therefore, the event organizer needs to list only eligible members to send the invitations. The members' information is given in a CSV file, and the record format in this data file is as follow:

```
firstName,lastName,YYYYMMDD
```

The new format that this office needs is displayed below:

```
Dear Mr/Mrs [lastName], [firstName]
```

You are asked to help them by writing a shell script `invite.sh` that accept a location of a CSV file that contains records of the original format, performs this conversion, and outputs the information in the new format. To simplify the task, simply subtract the birth year from the current year when calculating the age (ignoring the birth date and month).

Expected output

```
> cat guest.csv
James,Smith,19990123
Maria,Garcia,19700512
Jane,Hernandez,20031010
Peter,Wong,20000421
> ./invite.sh guest.csv
Dear Mr/Mrs Smith, James
Dear Mr/Mrs Garcia, Jane
Dear Mr/Mrs Wong, Peter
```

Task 5: Happy Countries (10 points)

Write a bash script called `happy_countries.sh` that lists out the names of the countries reported by 2018 World Happiness Report ranked by the happiness.

Your script will retrieve the information directly from the following wikipedia page, `https://en.wikipedia.org/wiki/World_Happiness_Report`. However, parsing the data directly from HTML is a headache. Luckily, you can request the page in an alternative format (raw wiki format) by using (`https://en.wikipedia.org/wiki/World_Happiness_Report?action=raw`). Notice the suffix `?action=raw`.

To keep things simple, you are to list only top 10 countries. For this task, you may only use the following tools: `curl`, `sed`, `awk`, `grep`, `tr`, `cut`, `sort`, `head`, `tail`.

Below is expected output of your script.

```
Finland
Norway
Denmark
Iceland
Switzerland
Netherlands
Canada
New Zealand
Sweden
Australia
```

Task 6: Git (10 points)

In the real world, developers use other developers' code for various reasons, such as testing ideas and improving specific operations. This assignment will explore this aspects of using git. In `forkit.txt`, save all line commands that you have used to complete this assignment in order that they have been executed. If any step asks you to use the Github website, there is nothing to save to the submitted file.

Follow the steps below to complete this task. Do not skip steps.

1. Sign up for an account on GitHub.
2. Log in and go to `https://github.com/KrityaLee/iccs207test`.
3. On that page, click on the Fork button to fork the repo to your account.
4. Back in your terminal, create a local clone of your fork.
5. Create a new branch called `foobar`.
6. Under `foobar`, append your full name as the last line of the file `week3/yadayada.txt`.
7. Commit the change to this branch.
8. Switch back to the master branch.
9. Append to `week3/yadayada.txt` with the text "Name added".
10. Merge `foobar` into the master. There will be a conflict. Resolve it by preseving both lines.
11. Complete the merge by committing and pushing the commit to your remote repo.

12. Create a pull request on `https://github.com/KrityaLee/iccs207test`, and fill out the body of the Pull Request with information about the changes you're introducing.

Your `forkit.txt` should contain lines of shell commands in the sequence that they are needed to complete the operation. For example, the sequence of shell commands below is typical for pushing a change to a remote server

```
git add blah.csv
git commit -m "Add two more topics."
git push
```

Notes: All operations must be performed through command lines, unless the instruction specifies otherwise. A step may require multiple commands.