ORIE 5270: Big Data Technologies
Homework 1 – Due: Monday, 03/08/2021

Instructions. For most problems in this homework, you will be asked to write a **shell script** that accomplishes a prescribed task. **Your script is not allowed to call / embed programs written in other languages**; for example, you might be tempted to implement everything using Python and then just call the appropriate python script via a bash script, but this is *not allowed here*.

You may use the internet to look up documentation for operating system and shell commands that may help you complete this assignment (in fact, you will probably need to do this anyway, and efficiently searching over online sources is a useful skill for both academia and industry). **Make sure you cite / acknowledge your sources** at the beginning of each file you submit.

The deadline to submit is **Monday, March 8th** at **12pm (noon) US EST)**. Submit your answers on **Gradescope**. Please submit a `.zip` archive containing a **single file per problem**.

Problem 1 (File summary)**.** Write a Bash script called `summary.sh` that summarizes a file. The script should receive one argument, which is the path to the file to be summarized – if none is specified, your program should (try to) process a file called `data.txt`.

Your script should generate another file named `<file-name>_summary.txt`, where `<file-name>` is the name of the input file to be summarized, with the following format:

```
<file_name>
Analyzed on: <current_date>
Number of lines: <number_of_lines>
Quick peek:
    <first_five_lines_of_file>
    ...
    <last_five_lines_of_file>
File Type: <type_of_file>
```

You might find these commands useful: `wc`, `head`, `tail`, and `file`.

Problem 2 (Find fields of certain type)**.** Suppose you are handed a `.txt` file that contains a number of lines in the format

```
<TYPE>: <Content>
```

Write a script called `fields.sh` that accepts two arguments: the first argument is the path to a `.txt` file following the above format and the second argument indicates the field of type `<TYPE>` you are interested in. Your script should then print the content of *all lines* starting with the given field but without any duplicates, and save the output to another file called `fields.txt`.

For example,

```
$ ./fields.sh data.txt AUTHOR
```

should find all lines in `data.txt` that look like

```
AUTHOR: <Name of author>
```

and generate a file called `fields.txt` that contains a list of *unique* author names.

Hint 1: use pipes to chain multiple commands.

Hint 2: to access arguments inside `bash` scripts, you can access the variables `$0`, `$1`, etc. See "Positional Parameters" in this url.

Problem 3 (Finding executables under a directory)**.** You just spent a long day at work (working on a machine shared with other people) prototyping a new machine learning framework. Unfortunately, you discover that several of the files that you generated might be executable by users that are not you (due to a mistake in setting permissions), leading to a potential security risk.

Write a Bash script called `whoops.sh` that accepts a single argument, which is the path to a folder containing the aforementioned files (note that the files themselves may be in subfolders). Your script should:

(i) Find all the files in that folder that are executable by other users (*Hint*: look up the `-perm` flag of `find`).

(ii) Change the permissions of the executable files in question so that other users can neither execute nor even read the executable files. (*Hint*: look up the `chmod` command).

(iii) Output the names of the executable files (one per line) in a file called `whoops.txt` (their names, but not their full paths; for example, if one of those files was `scripts/subscripts/python/run_ml.py`, the corresponding line in `whoops.txt` should only say `run_ml.py`).

Hint: see the `${var##Pattern}` construct in this tutorial.

Problem 4 (Github)**.** If you haven't already, navigate to Cornell's Github (`github.coecis.cornell.edu`) and activate your account. Then:

1. Create a fork of the `vc333/ORIE5270-SP21` repository through the web interface.
2. Clone your fork of the repository to make a local copy in your computer.

3. Create a new branch named after your netID and add a file called `<netID>.txt` (substitute your actual netID) containing your first name under the folder `hw1/`.

4. Push your changes to your branch with an appropriate commit message.

5. Submit a pull request to the original repo (`vc333/ORIE5270-SP21`). Follow the steps in this guide.

If your steps were correct, I should be able to merge your pull request with no conflicts. For this problem, you should submit a single `.sh` file containing all the `git` commands you had to write on the shell to complete this task.

Note: For Problems 5 and 6 below, you should submit a single `.txt` file per problem containing your answers to the questions. Include the `.txt` files in the `.zip` file you upload to Gradescope.

Problem 5 (Bash questions)**.** Answer the following questions about `bash` commands and constructs. When in doubt, consult the manual pages with `man <command>`!

1. What is the difference, if any, between the commands `touch file.txt` and `cat /dev/null > file.txt`? What about `cat /dev/null >> file.txt`?

2. Suppose you have defined a variable called `x` in the shell, but want to output the message `I like $x`. How can you accomplish that using the `echo` command?

3. Consider the following (seemingly identical) pieces of code:

```
# example1.sh
[ -f "test.txt" ] && echo "File exists" || echo "File does not exist"

# example2.sh
[-f "test.txt"] && echo "File exists" || echo "File does not exist"
```

   Try them on your `bash` shell and notice that something is wrong. Why is this happening? (explain in as much detail as you can)

4. Consider the following `if` statement:

```
# example.sh
if grep -i hello file.txt > /dev/null
then
    echo "Hello found!"
else
    echo "Hello not found!"
fi
```

Why do we redirect to `/dev/null` here?

5. What is the following piece of code doing?

```
while echo -n "Checking..."; who | grep "vasilis"
do
    echo "Still here"
    sleep 5
done
```

Problem 6 (More `git`). Answer the following questions about `git`. When in doubt, consult the manual pages using `man git`.

1. Suppose you are working on a project and want to create a new branch. Describe at least two different ways of creating a new branch using `git` commands.
2. You are working as a developer in a company that just started using `git`. Your manager wants to open-source a certain project in the following way:
   - There should be a stable version of your codebase available that is adequately tested and is only updated periodically with new features.
   - At the same time, users that want to try and experiment with new features should also be able to download code that your team is actively developing.
   - Finally, different parts of your team should be able to develop new features independently of each other, without interfering with other teams' work.

   Describe how you would use `git`'s branching features to facilitate the above development model (at a high level). For example: how many branches would you need? How would you manage each branch? Where would you merge each branch into?
3. If you want to undo some committed changes, it is common to use `git revert <commit_id>`. However, it will not work when you use it on a *merge commit* (a commit resulting after merging two branches). Why?