EXAM UNIX

- This is an examination. It is to be your work and your work alone.
- No exchange of information with another human entity in any form is acceptable.
- Reading google documents is ok. However, no exchanges via the internet are acceptable.
- It is ok to use your notes, your programs, your lab notes, and any other notes you have written for class.
- Using a textbook is ok.

After you download **Exam1-Fall022.zip** from Canvas, unzip the file if necessary, and a directory called **Exam1-Fall2022** will be created.

```
Follow these instructions in each file you submit
#mprocop2:09/29/2022:filename

#Q1
cat file

#Q2
ls -lt file

Shell error messages:
We will give partial credit in case of syntax errors
It is still preferred to comment out your code by using the # sign if
you don't know how to fix the error
```

```
Deadline and late penalty
The deadline is 8:30pm with 3 min grace period for you to check all the
scripts and submit them to the Gradescope assignment called Exam1.

Late penalty:
After 8:33pm, the penalty is 2 points for each late minute.

Gradescope Exam1 will close at 8:40m

IF YOU HAVE TECHNICAL ISSUES DURING THE EXAM LET THE PROFESSOR KNOW
```

Policy for cheating: sharing code on the exam is unacceptable and will earn you a 0 and take you straight to the ethics board. Do not use or share any study documents either!

General Grading:
* Following instructions (class header and Q&A pair)
* Full points for correct answer
* Partial points are available

For Cygwin and Ubuntu users: in each script that you submit, include:
 # Class header
 # Ubuntu or Cygwin user

Change into Exam1-Fall2022 and start your work in that directory.
If you open two terminals, make sure you change into Exam1-Fall2022 in both.

1. (35+1= 36) Make a file called **ex1-f22-system.bash** and in it answer the following questions. Answer the questions by using the appropriate commands.
You must start from the **Exam1-Fall2022** directory.

Q0. As a comment, include this sentence as Q0 for this script. Keep in mind that your exam **will not be graded** if you do not have this sentence as a comment.

 # I promise not to communicate with another human being in any way about this exam.

The Exam2-Fall2022 directory contains files and one directory. A simplified structure is reported here. There are files within the **files** directory, which are not reported in this scheme.

```
Exam1-Fall2022/files/
                script1.bash
                script2.bash
```

Q1.(8) Your current working directory is Exam1-Fall2022.
The **script1.bash** contains syntax errors, and the class header has the wrong jhed (it is not your jhed)

Q1.1 (2) Your first task is to understand what the errors are. For this do the following and use two lines of code.

- (1) view the contents of the script
- (1) run the script.

Q1.2 (6) Your second task is to fix both the syntax errors and the jhed (it should be your jhed) by using **one pipeline of bash commands** and to redirect the std output (which will be the fixed code and header with your jhed) into a file called **script1-fixed.bash.**

It is okay if the class header contains the script1.bash and not the script1-fixed.bash.
If you run the script1-fixed.bash, the output should look like that:

```
this is my location
/Users/mprocop2/Exam1-Fall2022 #this will change depending on your
computer
```

Q2. (3) Store the sentence reported below into a variable called **var**. Notice there are spaces and the # sign that are part of the sentence.

```
# script1.bash has been fixed #
```

Q3.(3) Your current working directory is Exam1-Fall2022.
Use variable **var** and append the sentence into the file **script1-fixed.bash**. Use one line of code.

Q4.(8) Your current working directory is Exam1-Fall2022.
Use one pipeline of commands to concatenate files file$1, file$2 and file$3 to obtain:

```
Hello Ciao Hola
```

Notice that there is one space between the words.

Files, file$1, file$2 and file$3 are within the **files** directory. Use relative path, and do not change current directory.
Do not hardcode the files file$1, file$2 and file$3 but use metacharacters.

In your pipeline use a bash command to translate:

`@` with `a`
`3` with `e`
`1` with `i`

Q5.(5) Your current working directory is Exam1-Fall2022.
Copy all the files (that are within the **files** directory) that have 3 characters total and do not END with either 1 or 2 or 3. Use metacharacters that match the description of the files. Copy them one directory back with respect to your current directory.
Do not change your current working directory and use relative pathnames.

Q6.(8) Your current working directory is Exam1-Fall2022.
Use two lines of code for this exercise.

- (1) Make this variable
  n1=2

- (7) Run the script script2.bash, redirect std error into the null devise, and process the std output by using a bash command with variable n1 to obtain:

```
A    1
B    2
I    3
L    4
I    5
T    6
Y    7
```

You should use variable n1 to specify the field number you sort by.

2. (21+1=22) Make a script called **ex2-f22-flow.bash** within your Exam1-Fall2022. You will run the script by providing a sequence of numbers from the command line, like this: `. ex2-f22-flow.bash 1 3 4 -10 30 40 -20`

In the script do the following:

- **Make one for loop** over a sequence of numbers provided from the command line, and for each number do the following:
  - Divide the number by 4, and store the result in a variable called **res**
  - **Within the loop test variable res** by including **one appropriate if-statement** that implements this logic.
    - If the result is greater than 0, implement a count for this case.
    - If the result is less than 0, implement a count for this other case.
    - Otherwise (if the result equals 0) print to screen that division results in 0. See the output reported below. You should include the same text as in the output below.

- After the for loop, display to screen:
  - the final count of results > 0 (greater then 0), and,
  - the final count of results < 0 (less then 0).

  You can use two echo statements. No need to use printf. See the output reported below. You should include the same text as in the output reported below.

If you run the script like that

```
. ex2-f22-flow.bash 1 3 4 -10 30 40 -20
```

the output should be:

```
1/4 results in 0
3/4 results in 0
3 results are > 0
2 results are < 0
```

Notice the > and < signs should be printed to screen.

3. (41+1=42) The data file **mtcars.csv** contains information about various cars, such as:

- model (1st field)
- mileage per gallon or mpg (2nd field)
- weight (7th field)

Make a script called **ex3-f22-cars.bash**, and in it answer the following questions:

Q1.(4) Generate a new file called **mtcars_data.csv** that does not contain the first line (which doesn't really contain the data, but only lists the fields). You can achieve this with either one command or with different commands of your choice, but do not hardcode any numbers.

If you do not know how to do this, in the next problems you can use the provided file **mtcars_data.csv.bak**.

Q2.(12) Sort the data file **mtcars_data.csv** according to the 2nd field (mileage per gallon or mpg) and print to screen only the 1st field (model), 2nd field (mpg) and 7th field (weight) of the sorted lines. The field separator between the outputted fields should be the colon :
Use redirection to store the std output into a file called **mpg_sorted.dat**.

This can be achieved in multiple ways, but use one pipeline of commands and redirection (one line of code).
Here, we report a few lines of the file **mpg_sorted.dat**

```
Cadillac Fleetwood:10.4:5.25
Lincoln Continental:10.4:5.424
Camaro Z28:13.3:3.84
Duster 360:14.3:3.57
……
……
```

If you do not know how to do this, or if your field separator is different than colon, you can use the provided file **mpg_sorted.dat.bak** for the next questions.

Q3. (12) Use a pipeline of commands to extract the first 5 mpg values of the sorted file **mpg_sorted.dat,** i.e., the 2nd field of the first 5 lines of **mpg_sorted.dat** and output those mpg values in one line (horizontally) (i.e., change a new line to a space). Use command substitution to store the mpg values in a variable called **mpg5**. Use one line of code.

If you do not know how to do it, for next questions make this variable:

```
mpg5='10.4 10.4 13.3 14.3 14.7'
```

No points awarded for making this variable.

Q4.(7) Use variable **mpg5** to calculate the average of the 5 mpg values and save the average in a variable called **av_mpg5** by using command substitution.

To calculate the average, you can use awk to do the sum and divide it by 5. Keep in mind that the mpg values are stored in a variable, and not in a file. Think about where to use the bash variable in your pipeline. Use only one line of code.

If you do not know how to do this, for next questions, just make this variable:
`av_mpg5=12.62`
No points awarded for making the variable.

Q5.(6) Do this part in two lines of code:

- (1) Make this variable

  `av_weight5=4.6858`

  which is the average weight of the corresponding mpg values.

- (5) Print to screen this sentence by using variables **av_mpg5** and **av_weight5.** Use printf in bash and do not use echo. You will need to format the numbers.

  ```
  The average weight is 5 and has mpg of 12.6
  ```

EC (3) Make a script called **EC-f22.bash** and in it calculate the average over each consecutive four values of mpg and weight in file **mpg_sorted.dat**. The output should be formatted to a single decimal place and should look like this:

```
12.1   4.5
15.0   4.0
16.2   3.6
18.5   3.4
20.2   3.0
21.8   2.7
25.1   2.6
31.8   1.8
```

Here, 1st line represents the average over 1st 4 lines (1-4) for mpg and weight, 2nd line is the average over 2nd four lines (5-8), and so on …
Do this **with a single while loop** and inside it calculates the average by using awk. Figure out how to calculate the sum of a numeric field in awk. It might also help to figure out how to use bash variables and select a line number in awk.
Your script should work on any data set with same fields as **mpg_sorted.dat** but different numbers of total lines.
This exercise shows that vehicles with low mpg have high weight.

Upload the following files to Gradescope Exam1:

ex1-f22-system.bash
ex2-f22-flow.bash
ex3-f22-cars.bash
EC-f22.bash

DO NOT SUBMIT ANY ADDITIONAL DATA FILES OR DIRECTORIES.