EXAM UNIX (90 POINTS + 3 EC)

- 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.
- It is ok to use your notes, your programs, your lab notes, and any other notes you have written for class.
- You can use all the material posted on the Canvas course.
- Using a textbook is ok.
- **Using internet to find out how to solve a problem is not allowed, including the use of chatGPT.**

After you download **Exam1-Spring2024.zip** from **Canvas**, unzip the file if necessary, and a directory called **Exam1-Spring2024** will be created.

Change into Exam1-Spring2024 and start your work in that directory.

```
Follow these instructions in each file you submit
●  Include the class header

●  Format of Q&A pairs
Example:

#mprocop2:02/21/2024:filename

#Q1
cat file

#Q2
ls -lt file
```

```
Deadline and late penalty
The deadline is 8:00pm with 3 min grace period for you to check all the
scripts and submit them to the Gradescope assignment called Exam1.

Late penalty:
After 8:03pm, the penalty is 2 points for each late minute.

Gradescope Exam1 will close at 8:10m.

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

Policy for cheating: sharing code on the exam is unacceptable as well as the use of internet to find out how to solve a problem (including chatGPT) and will earn you a 0 and take you straight to the ethics board. Do not share any study documents either!

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

Change into Exam1-Spring2024 and start your work in that directory.
If you open two terminals, make sure you change into Exam1-Spring2024 in both.

For Ubuntu users: in each script that you submit, include:
# Class header
# Ubuntu

1. **(35)** Make a file called **ex1-s24.bash** and in it answer the following questions, which refer to the directory structure depicted below.
   Answer the questions using the appropriate bash commands. You must start from the **Exam1-Spring2024** directory. The directory structure was already made within the Exam1-Spring2024 directory.

```
                       Earth.bash
                       Jupiter.bash
                       Mars.bash
                       Mercury.bash
                       Neptune.bash
                       Pluto.bash
                       Saturn.bash
                       Uranus.bash
              /planets/Venus.bash
              /receiver/key
Exam1-Fall2021/sender/$1
                      $2
                      $3
                      $33
                      $4
```

Q1. (2) Your current working directory is **Exam1-Spring2024.** List the contents of your current working directory, including hidden files.

Q2. (2) Change into the **sender** directory using a relative pathname.

Q3. (10) Your current working directory is **sender**. Within the **sender** directory there are some files. In this exercise, you will use only files named **$1**, **$2**, and **$4**.

With a single line of code perform the following tasks:

- vertically concatenate only the files **$1**, **$2**, and **$4** using a bash command and metacharacters.
- append the std output of the concatenation into a file called *key*. The *key* file is within the **receiver** directory. Use relative pathnames.

Partial points will be given for each task.
The content of the file *key* should be as follow:

```
#this is the encryption key
e   3
E   6
T   1
t   U
:   3
at  @
```

Note that the first line, beginning with '#', was already present in the *key* file.
The file *key* contains the encryption keys you will use later.

Q4. (2) Go back one directory using relative pathname.

Q5. (2) Use a bash command to display the current working directory.

Q6. (16) Your current working directory is **Exam1-Spring2024**. Within the **Exam1-Spring2024** directory, there is a hidden bash script. We will use the hidden bash script to send an encrypted message to a friend.

The hidden bash script contains two lines of code.
One line of code generates std error and the other line of code generates std output, which is the message that should be encrypted.

The encryption rules are:

```
e   3
E   6
T   1
t   U
:   3
at  @
```

This means that characters `eETt:` should be replaced respectively with characters `361U3` and the entire word `at` should be replaced with the character `@`

With one pipeline of commands do the following tasks:

- run the hidden script and redirect the std error into the null device.
- encrypt (translate) the std output of the bash script based on the encryption rules using only one tr and one sed command.
  Note that the single character t should be replaced with U, and the entire word at should be replaced with the character @

  The output of the pipeline of commands should be:

  ```
  m33U @ Uh3 M61
  ```

Partial points will be given for each step.
If you cannot identify the hidden bash script, you can use the provided bash script, message-bak.bash, for a loss of points for this part.

2. **(20)** Your current working directory is **Exam1-Spring2024**. Within the **planets** directory there are bash scripts named after planets, such as: Earth.bash, Jupiter.bash, Mars.bash, and so on. Each bash script contains bash code that prints the gravity of that planet to the screen. The gravity on Earth is designated as 1 G, and the gravity on the other planets of our solar system are in G's. Before starting this exercise view the contents of these bash scripts.

The script ex2-s24.bash is provided for you within the Exam1-Spring2024 directory, and it contains some code, which you should complete. <u>You cannot remove or change any provided code.</u>
The goal of this script is to generate a file called **planets.out**, which contains the planet name in the first field, the gravity value in the second field, and the field separator is a comma.

The script should do the following:

Q1 (13) Use **one for loop** over all the bash files that are within the directory **planets** using a metacharacter and a relative pathname.
Within each iteration of the loop:

- echo the loop variable and process the std output in pipeline to extract only the planet name and append the output (i.e. only the planet name) into a file called *first*. Do this using a single line of code.

  For example: one of the loop variable values is:
  `planets/Earth.bash`

  You should process the loop variable value in pipeline to extract only the planet name, i.e. `Earth`

- run the bash script using the loop variable and append the std output to a file called *second*.

Q2 (6) Once the for loop has completed its iterations, concatenate the files *first* and *second* and save the output of the concatenation to a file called **planets.out.** The file planets.out should be created within your current directory.
The file planets.out should contain the following:

```
Earth,1.0
Jupiter,2.36
Mars,0.377
Mercury,0.378
Neptune,1.12
Pluto,0.071
Saturn,0.916
Uranus,0.889
Venus,0.907
```

The output is not formatted, so <u>DO NOT FORMAT THE OUTPUT</u>.

Add additional code so that, if you run the script ex2-f24.bash multiple times, the contents of the **planets.out** do not change.

3. **(35)** The provided file **galaxies_raw.dat** contains information about rotation velocities measured for various galaxies.

Lines containing either > or # are comment lines.

The data are organized in the following fields:

- 1st frequency in MHz
- 2nd velocity in km/s
- 3rd flux density in mJy

First, examine the contents of the file to determine the field separator.

Make a script called **ex3-s24.bash**, and in it answer the following questions:

Q1. (6) Generate a new file called **galaxies.dat** that does not contain comment lines. Keep in mind that comment lines **contain** either ***** or >

You should generate the file **galaxies.dat** using a single line of code and using options of the grep command we did in class.

If you do not know how to do this part, for a loss of all points on this part, use the provide file **galaxies.dat.bak** for the following questions.

Q2. (13) In this exercise you will store in a variable the number of galaxies that satisfy certain conditions.
Use one line of code and to do the following:

- use awk to extract the lines of the file **galaxies.dat** with velocity (2nd field) greater than 8000 or velocity smaller than 4000 km/s, and in both cases density flux (3rd field) greater than 10.0.
- pipe the output of awk to a bash command to count those lines.
- perform command substitution to store that number into a variable. Choose a variable name that you prefer.

Q3. (3) Use echo with the variable you made in point Q2, and add text to display to screen the following:

```
The number of galaxies with velocity < 4000 and > 8000 is 138
```

Points will be deducted for unnecessary quotation marks.
However, alternative methods to remove the special meaning of < and > are acceptable.

If you did not create the variable in point Q2, create one and use it:

```
num=138
```

Q4. (12) Use a single **pipeline of commands** to display on screen the velocity and density flux of galaxies with density flux values (3rd field) less than -40.0. The output should be sorted from the largest to lowest velocity (2nd field). The output should also be formatted like this – at least 4 spaces between km/s and flux values. Those spaces are reported in grey. You must use the width to set those spaces.

```
v=9.89e+03 km/s     -44.75 mJy
v=9.88e+03 km/s     -46.64 mJy
v=9.87e+03 km/s     -48.03 mJy
```

EC (3) The EC is graded as all or nothing. For the EC you will make 2 scripts:

1st script.
Make a script called **CL.bash**.
The script takes 2 input arguments (2 integer numbers) from the command line like:
`CL.bash 1 10`

The script performs a test:
If the first number is greater than the second number:
    print to screen: *greater, here the difference:*
    print to screen: the arithmetic difference between the first and second number.
If the first number is smaller than or equal to the first number:
    print to screen: *smaller or equal*.

2nd script.
Make a script called **EC-s24.bash**.
The script uses a for loop to run the script CL.bash over pairs of numbers. The loop values should be set, so that in each iteration a pair of numbers (like 1 10) are the command line arguments to the script CL.bash.

You should think hard about the loop values in the EC-S24.bash. The loop values in this case must be written in this script, they are not provided from command line. You will run this script like this:
`. EC-s24.bash`

For example, if the following pairs of numbers are used as loop values:

```
1 10
20 2
30 30
3 40
```

you should get this output:

```
smaller or equal.
greater, here the difference:
18
smaller or equal.
smaller or equal.
```

This is all or None. To get all points you must submit the 2 scripts. In the two scripts there should not be any redundant code. If you use out of class material, you will lose all points.

```
Upload these files to Gradescope Exam1
ex1-s24.bash
ex2-s24.bash
ex3-s24.bash
EC-s24.bash and CL.bash
DO NOT SUBMIT ANY ADDITIONAL DATA FILES OR DIRECTORIES.
```