CS 279 - Homework 7

Deadline

11:59 pm on **Friday, November 4**

Purpose

To think about the `tee` command and the `BASH_REMATCH` array, and to get more practice with Bash arrays.

How to submit

You will complete **Problem 1** on the course Canvas site.

For the rest of the problems, you will create several files and then submit those to the course Canvas site.

NOTE: While I list the separate files you need to submit for each problem below, I am going to set up Canvas to *also* accept `.zip` files.

That is,

- you can submit each file to Canvas,
- OR, if you prefer, you may compress your files to be submitted into a single `.zip` file and submit that `.zip` file to Canvas.

Important notes

Assume, for all `bash` scripts in this course, that the following are required:

- Start each script with the line that is considered good style (and is a CS 279 course requirement), that specifies that this script should be executed using the `bash` shell
- After a blank line, put in one or more **comments** including at least the name of the shell script, your name, and its last modified date
- And follow these comments with a blank line.

Problem 1 - 10 points

Problem 1 is correctly answering the "HW 7 - Problem 1 - Short-answer questions on `tee` and the `BASH_REMATCH` array" on the course Canvas site.

Problem 2

Write a bash shell script `hw7-array1` or `hw7-array1.sh` that meets the following specifications. You should meet these specifications in order within your resulting script, BUT feel free to echo additional blank lines or "borders" as desired if you'd like your script's output to be more attractive.

- Part 1: write a Bash statement creating an array `stuff` containing at least 7 but no more than 10 elements of your choice, at least one element containing a blank surrounded by non-blanks. (For example, one element could be "`moo oink`".)
- Part 2: Then write a Bash statement that will now add a single array element to `stuff` with index `13` whose content is your first and last names, separated by a blank.
- Part 3: Then write a Bash statement that uses `stuff` to echo to the screen a descriptive message including the element in `stuff` with index `3`.
- Part 4: Then write a Bash statement that uses `stuff` to echo to the screen a descriptive message including

the size of `stuff` (the number of elements in `stuff`).

- Part 5: Then write a Bash statement that uses `stuff` to echo to the screen a descriptive message including the indexes of `stuff`.
- Part 6: Then echo to the screen a descriptive message saying that what follows are the elements of array `stuff`, one element per line, and finally write a Bash loop that will display the elements in `stuff`, one element per line.

Submit your resulting `hw7-array1` or `hw7-array1.sh` .

Problem 3

FUN FACT: you can initialize an array within a Bash script to the command-line arguments that script was called with using:

```
for_example=("$@")
```

(and the quoting here keeps command-line arguments with blanks from being split up into separate array elements)

Now adapt your `hw7-array1` or `hw7-array1.sh` into `hw7-array2` or `hw7-array2.sh`, making the following changes:

- Before Part 1: the script should complain and exit with a non-zero exit status if at least one command-line argument is not given.
- Modify Part 1: the initial contents of `stuff` should now be the command line arguments, instead of those you previously hard-coded.
- Modify Part 3: only echo the value of the element of `stuff[3]` if its length is non-zero -- otherwise, JUST set `stuff[3]` to a value of your choice
  - (Hint: its length can be zero if there are fewer than 4 command-line arguments OR if you actually give an empty string as a command-line argument -- the same test works in either case...)
  - (Hint: the syntax for getting the length of a particular array element is given in the Week 10 Lecture 1 projected notes.)

Submit your resulting `hw7-array2` or `hw7-array2.sh` .

Submit your resulting files:

- `hw7-array1` or `hw7-array1.sh`
- `hw7-array2` or `hw7-array2.sh`