

CS 279 - Homework 7

Deadline:

Due by 11:59 pm on **FRIDAY**, October 19.

How to submit:

Submit your files using `~st10/279submit` on nrs-labs, with a homework number of 7, by the deadline shown above.

Purpose

To practice with bash `if` statements, a variety of tests, checking exit status of a command, and interactive input, amongst other things.

Important notes:

- If you want to make additional "helper" bash shell scripts for use in one or more of the problems below, that is fine -- BUT be sure you submit them, also!
- **Each** bash shell script that you write is expected to include a descriptive opening comment block including your name and the last modified date.
- It is possible that your answers may be collected and posted to the course Moodle site.

The Problems:

Problem 1

Consider: what is a regular expression that will match a blank line in a file?

Create a shell script `strip-blank-lines.sh` that meets the following specifications:

- includes a descriptive opening comment block that includes both of your names and today's date
- it expects exactly one command line argument, expected to be a regular, readable file -- it should complain descriptively and exit with a non-zero exit status if this is not the case
- then, it creates a file `stripped-` followed by the name of the input file that contains the same contents as the input file EXCEPT with any blank lines stripped out
 - (or, if you prefer, it copies to the new file ONLY non-blank lines from the original input file)
 - If called with argument `pig.txt`, then, it would create output file `stripped-pig.txt` that contains `pig.txt`'s contents MINUS any blank lines.

Submit your resulting `strip-blank-lines.sh`

Problem 2

Fun fact: bash gives you a way to find out how many characters are in a string variable!

For variable `$myVar`, you can obtain its length using `${#myVar}`.

Use this in a little shell script `get-length.sh` which also meets these additional specifications:

- if more than one command-line argument is given, it complains in a descriptive message and exits
- if no command-line arguments are given, it asks the user to enter a string whose length is desired -- otherwise, it assumes that the single command-line argument is the desired input string
- it should output to the screen a descriptive message that includes the given string and the length of that string

Submit your resulting `get-length.sh`

Problem 3

Hm! It turns out, for all the things that are easy to test in a bash shell script, testing whether something is an integer takes a bit of a kluge!

It looks like one approach is to use a regular expression to do this. For our purposes in the next two problems, we in particular would like a non-negative integer, and we'll assume that it is not to be preceded by a `+`.

Write a little shell script `is-quant.sh` that meets the following specifications:

- Because we'll also use this in the next problem, we'd like it to "silently" work -- it won't output anything to standard output, it will simply exit with an appropriate error status
- if exactly one command-line argument is not given, it exits with a non-zero exit status of your choice
- otherwise, it should use a regular expression in an `if` statement to exit with an exit status of 0 (success!) if the input is indeed an (unsigned) integer greater than or equal to 0, and to exit with a non-zero exit status of your choice if the input is not.

Remember, you can test this by using:

```
echo $?
```

...after you try it out to see what exit status it produced.

Submit your resulting `is-quant.sh`

Problem 4

Something to note: you can call a shell script within another shell script. And once you have done so, you can look at `$?` to see the exit status of that call.

Write a shell script `make-line.sh` that expects exactly two command-line arguments, a string to repeat and a number of repetitions. It should simply output a **single line** of that string repeated that many times. That is,

```
$ make-line "^" 5
^^^^
```

`make-line.sh` should also meet the following additional specifications:

- if exactly two command-line arguments are not given, it should complain and exit
- it should use Problem 3's `is-quant.sh` to help it to verify that the second argument is a non-negative integer, and it should complain and exit if it is not.

Submit your resulting `make-line.sh`.

Problem 5

(adapted from University of Washington CSE 390, Spring 2010 Assignment 5: Basic Shell Scripting)

One more tidbit: one way (of several!) to do an arithmetic computation in bash is with a `let` construction:

```
$ myVar=3
$ echo $myVar
3
$ let myVar=$myVar+7
$ echo $myVar
10
```

Now we'll refactor `mantra1.sh` from Homework 6 - Problem 2 to be less trusting, and to give a little fancier output.

Write `mantra2.sh` that accepts two command-line arguments:

- a string for a message to print
- a number of times to print it

But `mantra2.sh` has a slightly different output than `mantra1.sh` -- rather than an opening and closing border line of a set length, you are expected to SURROUND your repeated message with a repeated character of your choice, and this border must exactly "fit" your message -- for example,

```
$ mantra2.sh "moo baa" 4
*****
* moo baa *
* moo baa *
* moo baa *
* moo baa *
*****
```

(except you don't have to choose `*` as your border character -- you may select the border character.)

And, `mantra2.sh` must also meet the following additional specifications:

- if exactly two command-line arguments are not given, it should complain and exit
- it should also complain and exit if the first argument's length is more than 76 (why 76? Because 80 is considered historically to be a reasonable line-length limit, and because the border makes the output message 4 characters longer.)
- it should use Problem 3's `is-quant.sh` to help it to verify that the second argument is a non-negative integer, and it should complain and exit if it is not
- it should use Problem 4's `make-line.sh` to create the borders

Submit your resulting `mantra2.sh`.