Bash Scripting Exercises

1. Write a simple Hello World script, but use printf instead of echo. Remember to check the man page of printf. In particular, note that in order to print a newline character, you have to add “\n” at the end of the string.

2. Write a “Hello ${USER}, the time is now…” script which says hello to the user who ran the script and prints the current time.

3. Create a script which takes as its input a number and a step-size, and counts from 1 to that number in increments of step-size. For example, given number = 50 and step-size = 11, the output should be

```
1
12
23
34
45
```

4. Yesterday, we downloaded a tar-file that contained lots of books from Project Gutenberg. Pick a book file from this collection. Write a script which takes as its input a search string, and outputs the first line in that book which contains the search string. You probably want to use **grep** for this, along with the **-m NUM** parameter.

5. Write a program which takes as its input two directory names. Return the total number of files in the two directories to stdout. (A simple way of counting files: Get **ls** or **find** to print all the files in the directory, and pipe it to **wc -l** to count the number of lines in the output).

6. Yesterday, we downloaded a tar-file that contained lots of book from Project Gutenberg, almost 100 files in total. All of them had cryptic file names. Let’s write a script which renames the files to the books’ titles.

First note that all the files have their titles close to the top of the files. If you use **head** (like cat, but only prints the first few lines of the file), you’ll find that **head pg996.txt** returns:

```
The Project Gutenberg EBook of Don Quixote, by Miguel de Cervantes

This eBook is for the use of anyone anywhere at no cost and with
almost no restrictions whatsoever.  You may copy it, give it away or
re-use it under the terms of the Project Gutenberg License included
with this eBook or online at www.gutenberg.net


Title: Don Quixote
```

And this goes for the other files as well.

It is possible to use **sed** to isolate the titles, using a **regular expression**. Regular expressions are fantastic tools for matching text and extracting information, and they are something that should be in any coder or data scientist’s tool belt. If you haven’t studied them yet, you are highly encouraged to do so. If you know them or are adventurous, you are highly encouraged to use them for your implementation. Pipe the result from grep into sed, and use a **capture group** to capture the title.

Otherwise, we’ll just use the observation that “Title: “ is 7 characters, and if we cut 7 characters away from the matching lines, we get the title. You can strip the first 7 characters from VAR through the syntax ${VAR:7}, i.e.:

```
MATCH=”$(grep …)”
TITLE=”$(MATCH:7)”
```

You might also find that your titles contain an undesireable character at the end, in which case you can strip away the last character by doing

```
MATCH=”$(grep …)”
TITLE=”$(MATCH:7:-1)”
```