Assignment 4: bash scripting exercise 2: `countmatches`

This course is a gateway into the bioinformatics concentration program at Hunter College in the Biology department or Computer Science Department. Eventually, you will be asked to write programs based on the context of information given. This exercise continues your practicing of bash scripting, and begins your journey into *thinking as a programmer*.

A ***DNA string*** is a sequence of the letters `a`, `c`, `g`, and `t` in any order. The letters are called ***bases***. For example,

`aacgtttgtaaccag`

is a DNA string with a length of `15` bases. Each sequence of three consecutive letters is called a ***codon.*** For example, in the preceding string, the codons are

`aac`, `gtt`, `tgt`, `aac`, and `cag`

If we ignored the first letter and started listing the codons starting at the second `a`, the codons would be

`acg`, `ttt`, `gta`, and `acc`

where we would ignore the last `ag`.

A DNA string can be hundreds of thousands of codons long, even *millions* of codons long, which means that it is infeasible to count them by hand.

It would be useful to have a simple script that could count the number of occurrences of a specific codon in such a string. For instance, for the example string above such a script would tell us that aac occurs three times and tgt occurs once.

Generally, we want to be able to find occurrences of arbitrary sequences of bases in a given DNA string, such as how many times `ttatg` occurs, or how many times `cgacgattag` occurs.

Instructions

Write a script named `countmatches` that expects at least two arguments on the command line.

- The first argument is the pathname of a dna file containing a valid DNA string with no newline characters or white space characters of any kind *within* it. (It will be terminated with a newline character.) This dna file contains nothing but a sequence of the bases `a`, `c`, `g`, and `t` in any order.
- The remaining arguments are strings containing only the bases `a`, `c`, `g`, and `t` in any order.

Error checking:
The script should check that the first argument is a file name, and that there is at least one other argument after it. If the first argument is not a file name or if it is missing anything after the filename, the script should output to the user

- the appropriate user error,
- a how-to-use-me message,

and then exit.
The script is not required to check that the file is in the proper form, or that the strings contains nothing but the letters `a`, `c`, `g`, and `t`.
The script is not required to check that the dna file contains a number of bases equal to a multiple of 3.

For each valid argument string, the program will search the DNA string in the file and count how many non-overlapping occurrences of that argument string are in the DNA string. To make sure you understand what non-overlapping means, the string `ata` occurs just once in the string `atata`, not twice, because the two occurrences overlap.

If your script is called correctly, it will output for each argument a line containing the argument string followed by how many times it occurs in the string. If it finds no occurrences, it should output `0` as a count.
For example, if the string `aaccgtttgtaaccggaac` is in a file named `dnafile`, then your script should work like this:

```
$ ./countmatches dnafile ttt
ttt 1
$ countmatches dnafile aac ggg aaccg
aac 3
ggg 0
aaccg 2
```

Warning: if it is given valid arguments, the script is not to output anything except the strings and their associated counts. No fancy messages, no words!

Testing: There DNA text files are in the cs132 course directory,

```
/data/biocs/b/student.accounts/cs132/data/dna_textfiles
```

to give to your script as the file argument.

Hint: You can write this script using `grep` and one filter command that appears in the course material. Although there are many filters commands, you do not need all of them to write the script. You have to read more about `grep` to know how to use it. The one filter command appears in the course material already.

Grading Rubric

This assignment is graded on a 100 point scale.

The script will be graded on its correctness foremost. This means that it does exactly what the assignment states it must do, in detail. Correctness is worth 70% of the grade. Then it is graded on its clarity, simplicity, and efficiency, worth 30% of the grade.

The objectives when writing any script, are

- *clarity* – it should be easy to understand by someone with a basic knowledge of UNIX
- *efficiency* – it should use the least resources possible
- *simplicity* – it should be as simple as possible

Submitting Requirements

- **Due date:** Sunday October 11, 11:59PM Eastern Standard Time
- Late submissions will be docked total points at the rate of 1 point for every day it is late. No submissions will be accepted after Sunday October 18, 11:59PM Eastern Standard Time.
- Accepted formats:
  - Screenshot image(s) in JPEG, PNG, or TIF of the script, its execution for a sampling various input and the resulting output of each execution.
  - A zip archive file containing the `countmatches` script.
    The steps to create a zip archive and to secure-copy files is outlined in the tutorials called '`zip` command for beginners' and '`scp` command for beginners', that are located in the Course Materials section on the course Blackboard.