Assignment 6: bash scripting exercise 4: `codonhistogram`

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 continues your journey into *thinking as a programmer*.

A DNA string is a sequence of the bases `a`, `c`, `g`, and `t` in any order, whose length is usually a multiple of three. In reality, it is not necessarily a multiple of three, but we will simplify it as such for discussion.

For example,

`aacgtttgtaaccagaactgt`

is a DNA string with a length of `21` bases. Recall that a sequence of three consecutive letters is called a codon. Assuming the first codon starts at position 1, the codons are

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

Those of you who know a little about genomics know that the open reading frame can be shifted to get a different set of codons. I want any of you who know this much to assume for discussion simplicity that there is only one open reading frame – the one starting at position 1.

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. Assuming the *central dogma of molecular biology*[1] holds true, a histogram of the *standard genetic code*[2] will prove useful in reporting the frequency of each recognized codon.

[1] https://en.wikipedia.org/wiki/Central_dogma_of_molecular_biology
[2] https://en.wikipedia.org/wiki/DNA_codon_table

Instructions

Write a script named `codonhistogram` that will accept the pathname of a dna file on the command line as its only argument.

Generally, the dna file should be a text 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 should contain nothing but a sequence of the bases `a`, `c`, `g`, and `t` in any order.

Error checking:
The script should check that it has one single command line argument, If it is missing the argument, the program should output to the user what the user error is, a how-to-use-me message such as

```
codonhistogram <dnafile>
```

and then exit.

The script should check that it is a file that it can read. If it canot, the script should output to the user the user error such as

```
codonhistogram: cannot open file <filename argument> for reading
```

where `<filename argument>` is the pathname of the dna file, a how-to-use-me message, and then exit.

The script should check that the dna file has only the letters `a`, `c`, `g`, and `t` and no other letter characters. If it does not satisfy this constraint, the script should output an appropriate user error message and then exit.

The script does not have to check that the file contains a number of characters equal to a multiple of three. If the file ends with a newline character, then the number of characters is equal to a multiple of three plus 1. If it does not satisfy this constraint, the script should output an appropriate user error message and then exit.

The script must count the number of occurrences of every codon in the file, assuming the first codon starts at position 1, and it must output the number of times each codon occurs in the file, ***sorted in order of decreasing frequency.*** For example, if `dnafile` is a file containing the DNA string `aacgtttgtaaccagaactgt`, then the command

```
./codonhistogram dnafile
```

should produce the following output:

```
3 aac
2 tgt
1 cag
1 gtt
```

because there are 3 `aac` codons, 2 `tgt`, 1 `cag`, and 1 `gtt`. Notice that frequency comes first, then the codon name.

Important: If two or more codons have the same frequency, your script should break the tie using alphabetical order of the codons. In this example, `cag` and `gtt` each occur just once, but because `c` precedes `g`, `cag` comes before `gtt` above.

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 25, 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 November 1, 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 `codonhistogram` 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.