CS 3360, Fall 2019
Professor Ward

Bash Scripting Assignment

You need to find a secret message buried in an innocent-looking list of seafood information. Fortunately you know how to extract it since you have intercepted an email with instructions:

> Greetings Agent X... to decode, take the lines which contain a four-letter fish name. Each fish name has a two part mock SKU number, e.g. 4443-069. You'll sort by the first part, in ascending order. If it's not odd, ignore it. Each line has a one character payload, which you get from the second part of the SKU by adding 3, and using that as the Ascii code for a character, for example, 97 for 'a', 98 for 'b', and so on. Then if it's a letter from a to z, look it up and replace it with the corresponding character in the codebook. Case doesn't matter. For example,
>
> hake 3-115
> snapper 5-219
> bass 181-99
> tuna 9-105
> char 007-106
> pike 846-723
>
> would, after filtering and sorting, give the sequence below, which would then map as follows:
>
> | | |
> |---|---|
> | 3-115 | 115 -> 118 -> v |
> | 007-106 | 106 -> 109 -> m |
> | 9-105 | 105 -> 108 -> l |
> | 181-99 | 99 -> 102 -> f |
>
> so if the codebook includes
>
> l e
> f p
> m t
> v u
>
> the final message will be "utep".

The test data above is found in **mini-fishlist.txt** and **mini-codebook.txt**. The actual file to decode is **innocuous.csv**; for this use **codebook.txt**. All are at the course homepage.

Part 1. [8 points] Use Unix commands to discover the secret message. Hint: **sort, join, awk, sed** and **grep** may be helpful. Please use mostly Unix commands; do not, for example, write a Python program to call.

Part 2. [8 points] Write a **bash** script that automates this process, taking as input a fishy file and a codebook, and outputs the secret message found using the rules above.

Hand in hardcopy including: 1) the secret message, 2) your **bash** script, and 3) a one-paragraph report noting any limitations, cleverness, or special features. Perfectionism is discouraged, and in particular your code need not be robust, as long as weaknesses are documented.

Due October 16