CSC 222 – 003 Winter 2022

HW2: Due 01/14/2022

1. (20 pts)
   a. Write a single-line shell command that outputs the words in /usr/share/dict/words that **begin** with an uppercase A **or** an uppercase E, followed by any two characters, followed by a lowercase b, and **ending** with a lowercase a **or** a lowercase n. Note that words greater than five characters in length are valid (i.e., there may be zero or more characters in between the lowercase b and the final lowercase a or lowercase n)! The command should **append** the output to a file in your home directory named **mywords**.

   b. List the words from the list that are outputted.

2. (80 pts) Write a C program (`tokenizer.c`) that repeatedly prompts the user for input at a simple prompt (see the sample output below for details). Your program should parse the input and provide output that describes the input specified. To best explain, here's some sample output:

```
:~$ ./tokenizer
$ ls -a -l -h
Line read: ls -a -l -h
Token(s):
 ls
 -a
 -l
 -h
4 token(s) read

$ clear
Line read: clear
Token(s):
 clear
1 token(s) read

$ exit
:~$
```

Note that the first and last lines are not part of program output (i.e., they are of the terminal session launching the program and after its exit). The program prompts the user (with a simple prompt containing just a **$ followed by a space**) and accepts user input until the command **exit** is provided, at which point it exits the program and returns to the terminal.

HW2: Due 01/14/2022

For all other user input, it first outputs the string **Line read:** followed by the user input provided (on the same line). On the next line, it outputs the string **Token(s):**. This is followed by a list of the tokens in the input provided, each placed on a separate line and indented by a single space. For our purposes, a token is any string in the user input that is delimited by a space (i.e., basically a word). The string ***n* token(s) read** is then outputted on the next line (of course, *n* is replaced with the actual number of tokens read). Finally, a blank line is outputted before prompting for user input again. The process repeats until the command **exit** is provided.

Hints:

(1) An array of 256 characters for the user input should suffice
(2) To get user input, `fgets` is your friend
(3) To compare user input, `strcmp` is your friend
(4) To tokenize user input, `strtok` is your friend

Make sure to comment your source code appropriately, and to include a header providing your name.

Plagiarized work will be subject to honor code penalty. The author(s) will get a zero on the assignment, at least.