CMSC246 Systems Programming - Lab 3

I/O Redirection and Arrays Practice

1. **HISTOGRAM:** Write a C program to print out a histogram from a set of input numbers. The input will be an undetermined number of scores on an exam (between 0..50, inclusive), one on each line, as shown below:

   ```
   47
   35
   50
   36
   48
   0
   48
   50
   42
   …
   ```

   The output will be a histogram as shown below:

   ```
   0: XXX
   1: XXXX
   2: X
   …
   48: XX
   49:
   50: XX
   The average score for 43 students is 37.23.
   ```

   Each X represents one occurrence of a score. Thus, in the above, there were three 0's in the input, two 48's, and no 49's, etc. The program also computes and prints out the average score, which is printed with exactly two digits after the decimal.

   A data file is provided (use: /home/gtowell/`Public246/Lab03/histo.txt`). Use Linux I/O redirection to read the data from the file. If your program is compiled into a.out, then you can use either:

   ```
   a.out < /home/gtowell/Public246/Lab03/histo.txt
   ```

   or

   ```
   cat /home/gtowell/Public246/Lab03/histo.txt | a.out
   ```

2. Extend the work for exercise 1 by capturing the histogram into a file. Show me the file.

3. **LETTER FREQUENCY COUNT:** Write a program to read a file of text (using standard input and redirection) and print out the frequencies of occurrence of each letter in the text, in ascending order. This is very similar to the first program, except you will now keep track of letters. Recall from class and your

text that you can use **char** values as numerical values. You will use that to index into an array of letter counts. You can use Linux commands to help you solve the problem. Count only letters.

for example the file `/home/gtowell/Public246/Lab03/michael.txt` contains the following text.
`Everyone can be taught to sculpt: Michelangelo would have had to be taught not to. So it is with great programmers.`
For which some of the output is:

```
A     9
B     2
C     3
……
X     0
Y     1
Z     0
```

Next, using the **sort** command, rearrange the output from your program to display the table in descending order of counts. So, assuming your compiled files is in a.out and the file is in file.txt you might use the command:

```
cat file.txt | a.out | sort
```

You will have to consult the man page for sort to determine how to sort this according to the numbers in the second column

Show me the sorted output for when run on the file `/home/gtowell/Public246/Lab03/Alice.txt`.