Project 1.1 Letter Histogram

Overview

A histogram is a bar graph showing the relative frequencies of various events. For this problem, you will create a program to plot a histogram of the relative frequencies of the various letters found in a plain text document. For the purposes of this assignment:

- Any character other than a letter is simply ignored.
- Case will be ignored (that is, the count for a letter will include both lower-case and upper-case versions).
- The histogram will be printed horizontally as a series of 26 lines, each containing a capital letter, a space, and a line of up to MAX_LENGTH asterisks representing the relative frequency of the letter in the text. By default, MAX_LENGTH is 50.

General Outline

Your program will need an array of 26 integers to hold the counts for the 26 letters. Obviously, the size of the array should be a symbolic constant created by **`#define`**. Simply read characters from standard input until **EOF** is detected, discarding non-letters,
and incrementing the count for every letter found. Note that standard input can be redirected from the

When the whole file has been processed, find the maximum count for any letter, say **max**. Compute the scale factor – the multiplier necessary to convert raw counts into a value in the range 0 through **MAX_LENGTH** – as follows:

(double) MAX_LENGTH / (double) max

Print the histogram lines, where the number of asterisks for each letter is the letter's count times the scale factor, rounded to the nearest integer.

Notes

1. The files distributed with this project include skeleton source code, a Makefile, a README file, and sample input and output files. Download then from this zip file : Project1-1.zip
2. You'll need to include **stdio.h** and **ctype.h** to gain access to the I/O routines and character classification functions and macros you'll need. You are encouraged to view these files, which are in directory **/usr/include** on linus
   or when using cygwin, as they declare interfaces to a variety of useful functions and macros.
3. Your program must be formatted in accordance with the style illustrated in Kernighan & Ritchie. Style includes placement of braces, indentation conventions, etc.
4. Comments preceding functions must be of the following form:

```
/*
* Brief description of the function, its arguments,
* and the value returned (if any).
*/
```

4.

Your program must use meaningful names for functions and variables. Function arguments and local variables can use somewhat shorter names than those of global variables accessed by many functions.

5. Your program must use symbolic constants for all non-obvious values (in general, any value other than -1, 0, or 1 is non-obvious).
6. Remember that standard input can be redirected from the keyboard to a file by:

```
$ ./histo < SelectedInput.txt          /* executable histo gets input form
SelectedInput.txt */
```

7. Remember that standard output can be redirected from the screen to a file by:

```
$ ./histo < SelectedInput.txt > SelectedOutput.txt
```

Deliverables & Due Date

- **Project 1.1 is due at the end of the day (11:59 PM) Friday of Week 4, (-15% if less than 24 hrs late, -30% by start of Week 5 (12:01 AM Monday), no submissions accepted after that time)**
- Submit one source file named ‘project1-1.c” to the **Project 1-1 *Code*** drop box in myCourses.
- Submit the following artifacts to the **Project 1-1 *Doc*** drop box in myCourses.
    1. Project log including estimated time / actual time, design notes, testing strategy (one text document).
    2. Any test code or scripts that you used in testing your solution.