Assignment Homework 2

Due Date Committed by 4/10/2017, **before class**.

Changelog

Grading

Your homework will be graded on the following items:

10	Comments & Documentation: is your code well documented?
10	Clear & Concise: is your code easy to understand?
10	Warnings: is your code warning free and portable?
10	Correctness: does your code do the right thing?
10	Implementation/Algorithm: manner in which you solve the task.
10	C Quality: proper use of language constructs, error checking.

Goal

The goal of this homework is to become familiar with editing and writing small (single file) C programs and with the basic functions for file I/O.

Character Histogram

Write a program (`histogram.c`) which reads the file **given to it on the command line** and writes to the screen a histogram showing the distribution of characters found in the input file. In the histogram, *skip character classes which have zero frequency.* Case does not matter (i.e. 'A' is the same character as 'a'). The following classes should be counted: A–Z, 0–9 (each letter or digit is to be counted separately).

Output (for `./histogram histogram-in.txt`, **first 10 lines only**):

```
1: # (1)
3: ## (2)
4: # (1)
5: # (1)
6: ## (2)
8: # (1)
a: ################################################################################### (83)
b: ############# (13)
c: ############## (14)
d: ########################################## (42)
```

Your program should **exactly** follow the same format output as the example above.

Commit the following files: `histogram.c`, `Makefile`, `histogram-out.txt` in directory `homework/hw2/histogram`.
(`histogram-out.txt` is the output of your code when ran on `histogram-in.txt`.)

Hints:

- look in the man pages for `stdio.h` and `ctype.h`.
- The lecture 2 materials contain some good starting points: the `Makefile` as well as the `header_protect.c` example program, which shows how to access command line arguments.
- To send the output of your program (any program) to a file instead of to the screen you can *redirect* the output of your program by invoking it as follows:
  `./histogram > histogram-out.txt`.

- Make sure your program compiles warning free *under strict C11 standard rules* using `gcc` on `linux.cs.uchicago.edu`.
- See the `file.c` for example code on how to access files. The manpages for the corresponding functions are very helpful as well!

Scrambling Words

Write a program which reads the file named on the command line and prints to the screen a scrambled version of the words found in the file (words are separated by newlines or by blank characters (see `isblank()`). Output one scrambled word per line. You can assume lines will not be longer than 1024 characters, and that words will not be longer than 128 characters. You can ignore any non-alphanumeric character (i.e. the word '`pepper,`' becomes '`pepper`', the word '`Bistritz.--Left`' becomes '`BistritzLeft`').

Example output (for `./scramble histogram-in.txt`, **first 10 lines only**):

```
nnJhtoaa
eksHrra
nloraJu
3
ayM
LeistzBrtfit
uhMcni
at
853
MP
```

Commit the following files: `histogram.c`, `Makefile`, `histogram-out.txt` in directory `homework/hw2/histogram`.
(`histogram-out.txt` is the output of your code when ran on `histogram-in.txt`.)

Commit the following files: `scramble.c`, `Makefile`, `scramble-out.txt` in directory `homework/hw2/scramble`.

Hint:

- look in `stdlib.h` (manpage) for a function which can provide you with *pseudo-random* numbers.
- You probably want the '%' operator.
- In your `Makefile`, make sure to record all dependencies!
  (i.e. if a `.c` file depends on a header (you can ignore system headers), `Make` should know this!)
- Regarding file access:
  - Do not forget to close the file...
  - Make sure to check for I/O errors!
    (You can output an error and terminate the program if an I/O error occurs.)