In linguistic applications, it is often helpful to analyze text for various properties. One such analysis computes a letter frequency histogram for a given text file. This is useful, for example, in solving puzzles like cryptograms where commonly occurring letters are likely to map to letters in the set {R, S, T, L, N, E}. It's also useful in designing word games, such as Scrabble™ in which letter scores are higher if they occur infrequently in common words.

This assignment explores the function and implementation of basic conditional execution, iteration, and memory access mechanisms to realize a letter frequency histogram. You will write two programs, first in C and then in MIPS. In each program, the case-insensitive frequency of occurrence of each letter of the alphabet is computed (in other words, for each letter, compute the number of times it occurs either in upper- or lowercase). For example, in the histogram for "This is 1 short text." all letters have a count of 0, except for the following [E: 1, H: 2, I: 2, O: 1, R: 1, S: 3, T: 4, X: 1]. The program does not count numbers, punctuation, spaces, etc.; it only counts letters.

HW2-1: In the C program, the letter frequency will be displayed (on the screen), along with the usage percentage for each letter. The program should work with an arbitrary length text file. A shell program, `HW2-1-shell.c`, is provided to help you get started. Rename the shell file to `HW2-1.c`. You should use `gcc` under Linux to develop your program. You should compile and run your program using the Linux command lines:

```
> gcc HW2-1.c -g -Wall -o HW2-1
> ./HW2-1 address.txt
```

Output for the sample data file `address.txt` is provided in `expected-output-HW2-1.txt`. Your program's output should match this format and values *exactly* to maximize credit in the automatic grading process.

HW2-1 Deliverable: Name the file **`HW2-1.c`** and submit the file on T-square by **5:00pm on Friday, 30 January 2015**.

Hints: the function `getc` may be useful in reading each character from the input file. The ASCII codes for letters of the alphabet can be found at http://www.ascii-code.com/.

HW2-2: In the MIPS program, the letter frequencies will be computed and stored in the 26 element vector in memory, starting at label `Letters`. The input to the MIPS program will be the 64 data words in memory, starting at label `Text`. Each data word contains 4 bytes, each byte is an ASCII code. These 64 four-byte words contain the ASCII codes for the first 256 characters in `address.txt`. The end of text is denoted by the label `End:`. An assembly language shell program, `HW2-2-shell.asm`, is provided to get you started. *When your program completes, the letter counts should be stored in the* `Letters` *array.* (These should match the output shown in `expected-output-HW2-2.txt`.) For the MIPS part, you do not have to compute the usage percentages.

Hints: Note that the Misasim memory implementation is little endian (the least significant byte is located at the word address). To view the ASCII codes more easily, select the hexadecimal display base under Options in MiSaSiM. In MIPS, the load byte instructions may be helpful (http://www.ece.gatech.edu/academic/courses/ece2035/assignments/Load-Store-Byte-Insts.pdf).

HW2-2 Deliverable: Name the file **`HW2-2.asm`** and submit the file on T-square by **5:00pm on Friday, 30 January 2015**. The starting *shell* program should not be modified except for the replacement of the comment `/* your program goes here */`. Your program must return to the operating system via the **`jr`** instruction. *Programs that include infinite loops or produce simulator warnings or errors will receive zero credit*.

All code (MIPS and C) must be documented for full credit. You should design, implement, and test your own code. **Any submitted project containing code not fully created and debugged by the student constitutes academic misconduct.**