Individual project.

**Deadline:** 11 PM, Monday, Dec. 9, 2013. Submissions won’t be accepted after this deadline.

The assignment has two parts (Parts (a) and (b)). Both parts must be done in MAL.

The source files for the two parts must be named `p5a.mal` and `p5b.mal` respectively.

The two source files must be submitted *together* using the `turnin-cs333` command.

README file will be available by 10 PM on Saturday, Nov. 23, 2013.
Important Remarks:

- There is no two-day grace period for this program.

- When you use `turnin-csi333` to submit your files, you must submit both `p5a.mal` and `p5b.mal` at the same time. Thus, the Unix command to be used for turning in your files is the following:

  `/usr/local/bin/turnin-csi333 -c csi333 p5a.mal p5b.mal`

- Programs for which the `spim` system reports syntax or runtime errors won’t receive any credit.

- Your program must work correctly on `itsunix.albany.edu`. 
**Overall goal:** To provide practice in writing programs in MIPS Assembly Language.

**Weightage:** 10%

**Total Points:** 100

- **Part (a):** 40 points (34 points for correctness and 6 for documentation).
- **Part (b):** 60 points (52 points for correctness and 8 for documentation).

**Caution:**
In MAL, comments start with ’#’. You **cannot** use "/*" and "*/" for comments.
Description of Part (a)

Remarks:

- A **positive** integer is one that is *strictly* greater than zero.

- An integer $x$ **evenly divides** an integer $y$ if the remainder when $y$ is divided by $x$ is zero.

- In MIPS, each integer is stored using 32 bits.
**Goal:** To read a positive (decimal) integer and compute the following values:

- The number of 1’s in the *right half* of the binary representation of the integer.
- The number of 0’s in the *left half* of the binary representation of the integer.
- The highest power of 2 that evenly divides the integer.
- The value of the largest digit in the decimal representation of the integer.
Example: Consider the decimal integer 1536.

- The 32-bit binary representation for 1536 is:
  0000 0000 0000 0000 0000 0110 0000 0000
  (The 16 bits in the left half are all zero.)

- The number of 1’s in the right half = 2.

- The number of 0’s in the left half = 16.

- Since $1536 = 3 \times 2^9$, the largest power of 2 that evenly divides 1536 is 9. (This value 9 is also the number of 0’s at the end of the binary representation of 1536.)

- The value of the largest digit in the decimal representation = 6.
**Program Outline:**

1. Prompt the user for a positive integer.

2. Read the integer.

3. Compute the four quantities mentioned above and print the answers.

4. Stop.

**Note:** Each time your program for Part (a) is executed, it should handle just one integer.
unix2> /usr/local/bin/spim
.
.  <--- Initial lines printed by spim.
.
(spim) read "p5a.mal"
(spim) run
Positive integer?  1536
No. of 1's in the right half = 2
No. of 0's in the left half = 16
Largest power of 2 = 9
Largest decimal digit = 6
(spim) quit
unix2>
unix2> /usr/local/bin/spim

.  
.  <--- Initial lines printed by spim.  
.

(spim) read "p5a.mal"
(spim) run
Positive integer? 123
No. of 1’s in the right half = 6
No. of 0’s in the left half = 16
Largest power of 2 = 0
Largest decimal digit = 3
(spim) quit
unix2>
You can assume that the user will type a positive integer. No error checks are needed.

There is *no need* to convert the integer to binary; when the integer is read in (using `syscall`), it is already in binary form.

Use bitwise operations to count the number of 1’s (0’s) in the right (left) half.

To find the highest power of 2 that divides the integer, count the number of 0’s at the end of the binary representation or use successive divisions by 2.

To extract the decimal digits and compute the largest digit, use successive divisions by 10.
**Goal:** To read a line of text and output the following information:

- The number of non-whitespace characters in the line.
- The number of words in the line.
- The maximum length of a word in the line.
- The minimum length of a word in the line.
- The word of maximum length in the line.
- The word of minimum length in the line.
Remarks:

- A **whitespace** character refers to a space, a tab or the newline character.

- A **word** is any sequence of characters that does not contain a whitespace character.

- Assume that the line typed by user has at most 80 characters, including the ‘\n’ character. (There is no need to check this condition.)

- If there are two or more words of maximum length in the line, then the program should print the word of maximum that appears *first* in the line.

- A similar comment applies to the word of minimum length.
Example: Suppose the input line is:

This example contains five words.

- No. of non-whitespace characters: 29
- No. of words: 5
- Maximum length of a word: 8
- Minimum length of a word: 4
- Word of maximum length: contains
- Word of minimum length: This

Note: In the above example, there are two words of minimum length (namely, "This" and "five"). The word that occurs first is "This".
Outline for Part (b):

1. Prompt the user for a line of text.

2. Read the line of text.

3. If the line has only whitespace characters
   Print a suitable message and stop.
   Else
   Compute the required quantities, print the answers and stop.

Notes:

- Each time your program for Part (b) is executed, it should handle just one line of input.
- Except for checking for a blank line, no error checks are needed.
unix2> /usr/local/bin/spim

.  
.  <--- Initial lines printed by spim.
.

(spim) read "p5b.mal"
(spim) run
Text? A short line.
No. of non-whitespace characters: 11
No. of words: 3
Maximum length of a word: 5
Minimum length of a word: 1
Word of maximum length: short
Word of minimum length: A
(spim) quit
unix2>
unix2> /usr/local/bin/spim

.  
 .  Initial lines printed by spim.
 .

(spim) read "p5b.mal"
(spim) run
Text? This example contains five words.
No. of non-whitespace characters: 29
No. of words: 5
Maximum length of a word: 8
Minimum length of a word: 4
Word of maximum length: contains
Word of minimum length: This
(spim) quit
unix2>
Remarks and Suggestions for Part (b)

- For Part (b), your MAL program must have **at least one function in addition to the main program**.

- Study Lecture 15 (in particular, the material on arrays of character in MAL) before working on Part (b).

- You may find it useful to write a function that returns information (e.g. starting and ending indices) about the next word.
Programs will be graded using a script written by the TAs.

The script will compile your source program, generate the executable version and run the executable on new test data.

The TAs will grade the version that you submit; once the submission is closed, you won’t be allowed to make any changes to your program.

You must follow the programming and documentation guidelines indicated in “Course Policies”.