CS1112 Fall 2013 Project 5 due Monday, 11/11, at 11pm

You must work either on your own or with one partner. If you work with a partner you must first register as a group in CMS and then submit your work as a group. *Adhere to the Code of Academic Integrity.* For a group, "you" below refers to "your group." You may discuss background issues and general strategies with others and seek help from the course staff, but the work that you submit must be your own. In particular, you may discuss general ideas with others but you may not work out the detailed solutions with others. It is not OK for you to see or hear another student's code and it is certainly not OK to copy code from another person or from published/Internet sources. If you feel that you cannot complete the assignment on you own, seek help from the course staff.

Objectives

Completing this project will solidify your understanding of character array and cell array. You will also work with a text file and create graphics.

1 The Game "Hangman"

This is an old game that sadly has a rather unpleasant name (and matching graphic). Another name for the game is "Gallows", which is also of questionable taste . . . This game often is used for learning the vocabulary of a language. If you don't know the game, play a game or two on http://www.manythings.org/hmf/8997.html. Next, check out the demo video, available on the course website, of our "Hangman" game.

Our version of "Hangman" uses words chosen from a list of commonly used English words. A round of the game begins with a randomly chosen word (hidden from the player except for the length). The player keeps guessing until *seven* wrong guesses have been made or until the word is completed, whichever happens first. At the end of a round, the user is prompted to indicate whether he/she wants to play again. The game ends when the player indicates that he/she wants to quit. You will build this program in a modular fashion by writing four functions:

- `hangman`: the main function that starts the game, handles user interaction, and creates and updates the figure. This function calls the remaining three functions.
- `getWords`: read the data file and return the words in a cell array
- `newWord`: randomly select a word that hasn't been used before from the cell array
- `findLetterInWord`: locate a letter in a word

You may write subfunctions as appropriate. You can use the general built-in functions that we have used in the past such as `length`, `rand`, `plot`, . . . , etc., but only the following *file* and *string* handling functions are allowed: `fopen`, `fclose`, `feof`, `fgetl`, `char`, `str2double`, `strcmp`, `lower`, `upper`, `isletter`. Additionally, there are two general built-in functions introduced in *Insight* that you may want to use: `all` and `any`. Use `help` in MATLAB if you want to learn more about them. Also, be sure to read the 1-page document *File i/o example*, posted along with the notes of Lecture 18; it will help you with this project.

Do *not* use functions `find`, `strfind`, or `findstr`.

1.1 Getting the data

The file `popularWords.txt`, available on the course website, contains more than 5000 popular words in the English language. Here're the first few lines in the file:

```
Most popular words in the English Language
From Englishclub.com
columns 1-5 is the word number
columns 9 and on contains the word

00001   the
00002   be
00003   and
00004   of
```

The words of interest appear on the "numbered" lines (starting with the line numbered '00001' that contains the word 'the' in this file); the other non-numbered lines are considered the file header and should not be used as word data for the game. You can see the whole file by using any plain text editor (such as Notepad) or even MATLAB (just double-click on the filename in MATLAB). In our game of Hangman, we use only the words that do not contain any punctuation marks (hyphen, apostrophe, etc.) and that are at least five letters long. Do not change the case of any letter from upper to lower or vice versa. Implement the following function as specified:

```
function C = getWords(fname)
% fname is a string that names a plain text file.
% Read from the file and store in a 1-d cell array C all the words that are
% longer than 4 letters and that do not contain any punctuation marks.
% Exclude the "header lines" in the file--use only the words on the
% numbered lines.
```

The data file may contain several occurrences of the same word; put all of them in the cell array `C` as long as they meet the criteria (longer than length 4 and letters only). There is no need to check for repeated words. This function should work with any data file that is in the format as noted above and in the header lines of `popularWords.txt` (word data stored on "numbered" lines only with the given column count).

1.2 Selecting a word

Implement the following function as specified:

```
function [word, updatedUsedWords] = newWord(C, usedWords)
% word is randomly chosen from cell array C but is not in cell array usedWords.
% C is a 1-d cell array of strings; C is not empty.
% usedWords is a (possibly empty) 1-d cell array of strings.
% word is a string randomly selected from C; word is not in cell array usedWords.
% updatedUsedWords is a cell array of strings; it is usedWords with one extra cell
%   containing word.
% Assume C contains many more different strings than usedWords does.
```

Recall that `C` may contain the same word in multiple cells; you need to make sure that the returned `word` is not identical to any string in `usedWords`.

1.3 Searching for a letter in a word

Implement the following function as specified:

```
function [found, tf] = findLetterInWord(let, word)
% Locate a letter (let) in a word (word), regardless of case.
% let is a char scalar.  word is a 1-d char array (string).
% found is 1 if the letter is found; otherwise found is 0.
% tf is a vector that has the same length as word; for each valid index k,
%   tf(k) is 1 if word(k) is let, regardless of case; otherwise tf(k) is 0.
```

Consider this example: `let` stores the scalar `'a'` and `word` stores the string `'American'`. Then the function returns in `found` the value 1 (true) and in `tf` the vector `[1 0 0 0 0 0 1 0]`.

1.4 Putting together the game

By making effective use of the above functions, implement function `hangman` as specified:

```
function hangman(fname)
% Run the game hangman using words that are chosen from the file named by
% the string in fname.
% The game begins with the gallows drawn and dashes are displayed above the
% gallows to indicate the number of characters in the word.  Each time the
```

```
% user guesses an incorrect letter, a body part is added to the figure.  A
% round of the game ends when the user has made seven incorrect guesses or
% when the user has guessed the word correctly, whichever occurs first.
% Throughout the game, display the word status in the title area of the
% figure window and update the hangman figure as appropriate.
% When a round ends, display in the title area of the figure window a
% message indicating the word and whether the round is won or lost.  In the
% Command Window, prompt the user about whether to play again.
% Words should not be repeated in a game; you can assume that the user
% chooses to play again far fewer times than there are available words.
```

Some additional considerations and specifications:

- Note that the maximum number of wrong guesses is seven but there is no restriction on the number of correct guesses. After each of the first six wrong guesses, one of these body parts should be added to the figure: head, body, left arm, right arm, left leg, and right leg. The seventh wrong guess changes the color of the drawn stick-figure man and ends the round. (If the user repeatedly enters the same *correct* letter before the word is complete, the round will just keep going since correct guesses are not counted. However, repeatedly entering the same *wrong* guess will add to the count, and diagram, each time.

- Use the figure window to display both the diagram and the word status. The word status, to be displayed using the `title` command, is the current word, with the not-yet-guessed letters displayed as dashes and the correctly guessed letters filled in. You must respect the case of the chosen word. For example, if the word is `American` and the user correctly guessed the letter `'a'`, then the displayed word status should be `A-----a-`
  You can increase the font size to, say, 16 points (default is 10) using the *FontSize* property like this:
  ```
  message='Winner!'; title(message, 'FontSize', 16)
  ```
  To clear the axes of plotted items at the beginning of each round, use the command `cla`.
  If you use the long ago given functions `DrawRect` and `DrawDisk`, there is no need to submit them.

- Use the Command Window to obtain user input. At the beginning of each round, clear the command window using the command `clc` and display a message indicating the number of characters in the current word to be guessed. When prompting for a user input string (or character), use a second argument, `'s'`, in the `input` built-in function to indicate that the expected user input is a *string*, like this:
  ```
  let= input('Guess a letter :  ', 's');
  ```
  This way, the user does not need to put quotes around the letter that he/she enters.

- Optional things to add, for convenience and/or fun ...
  (1) You may want to "dock" the figure window into the MATLAB desktop so that you can see both the figure and the command window at the same time. You can click the docking down-arrow for each game, or easier is to specify it in the `figure` command, like this: `figure('WindowStyle','docked')`
  (2) You can add the guessed letters to the figure window, under the gallows, so that the user doesn't have to scroll up in the Command Window to see which letters he/she has guessed previously. Use the `text` command. Here's an example:
  ```
  usedLetters='a e i r'; x=0; y=0; text(x,y,usedLetters)
  ```
  You can additionally use the *FontSize* property in `text` as you do in `title`.

Have fun playing the game! Then submit your files `hangman.m`, `getWords.m`, `newWord.m`, and `findLetterInWord.m` on CMS.