COSC 1336 Lab: 11

Relevant reading: Chap. 9, Sec. 11.1-11.3

Due: Dec. 4, 2:30 pm

(Late date: Dec. 11, 2:30 pm)

50 Points

Problem 1. [**50 points**] Download the files `hangman.py` and `words.txt`. For this problem, you will write most of a Hangman game. If you aren't familiar with the game of Hangman, I recommend you go read the relevant Wikipedia article before you begin. Write each of the functions specified below. I recommend you read through the whole problem before you begin, so you know where you are headed. There is a "Hints" section at the end with some tips on getting each part done. I recommend trying to do the assignment without looking at the hints, but if you get stuck, they might be helpful.

a. [**15 points**] Write a function called `replace_all` that receives three things, and returns two. It receives three strings: one, which I'll call `word`, is a single word; another, which I'll call `so_far`, is the same length as `word` but contains dashes in the positions that have not yet been correctly guessed; and lastly `letter` is a single-character string. The function should return a Boolean indicating whether there were any instances of `letter` found in `word` and a copy of the string `so_far` with the letters in the locations where `letter` was found in `word` replaced with the actual letter. The following Python interpreter transcript shows how the function should behave when it is run.

```
>>> replace_all("apple", "-----", "p")
(True, '-pp--')
>>> replace_all("apple", "-pp--", "j")
(False, '-pp--')
>>> replace_all("apple", "-pp--", "e")
(True, '-pp-e')
```

Make sure you test the heck out of this function before you move on! Remember that there are hints at the end of this document if you are stuck.

b. [**15 points**] Now write a function called `play` which receives a single string variable, `word`, and plays a round of Hangman with that word as the word that the user should guess. Here's a transcript of how this function should behave:

```
>>> play("apple")
Guess this word: -----
Enter a letter: p
Guess this word: -pp--
There is no 'j' in the word.
 ----
|/   |
|
|
|
|
^
Guess this word: -pp--
Enter a letter: f
```

```
There is no 'f' in the word.
 ----
|/   |
|    O
|
|
|
^
Guess this word: -pp--
Enter a letter: e
Guess this word: -pp-e
```

This should continue until either the word is guessed or the number of allowed guesses is exceeded. You should use the `draw_stick_figure` function to create the drawing each time an incorrect letter is guessed. Right now, it takes eight guesses to fully draw the stick figure, so the maximum number of guesses the user can have is eight. You are welcome to modify the `draw_stick_figure` function to allow for more guesses if you'd like.

When either the maximum number of guesses is exceeded or the word is guessed, an appropriate message should be displayed. If the word was not guessed correctly, the user should be told what the word was.

c. [**12 points**] Now write a `main` function that allows the user to play the game repeatedly, getting the words from the file `words.txt` and passing them to the `play` function you just wrote. After each `play` function call returns, the user should be given the opportunity to stop playing the game. The game should stop if either the user declines to continue playing, or the `words.txt` file is exhausted.

d. [**8 points**] Once the basic game is working, modify the `play` function so that it keeps track of the user's guesses, and if she guesses something that was already guessed, simply discards the new guess (and tells the user she already guessed that letter). This should be a fairly minor modification to the function, so if you are thinking of completely dismantling and reconstructing your function, you should probably talk to me about your plan beforehand.

Hints

a. The `replace_all` function:

- I recommend you use the `string find` function to find the index of each instance of `letter` in `word`. Notice that `find` has a second optional parameter that allows you to specify the index where you want to start the search. So, once you've found the first instance, you'll want to start the next search one place further on.
- The `string replace` function isn't as useful as you might hope for this function, as it replaces a given substring with a different substring, but we always want to replace a `-`, and typically not all of them. Instead, one way to do this is by slicing and concatenating. So, for example, if the variable `index` indicates the location of a letter we'd like to replace with the letter `'p'`, say, we might write the following expression:

  ```
  so_far = so_far[0:index] + 'p' + so_far[index+1:]
  ```
- You'll need to have a Boolean flag variable to keep track of whether or not an instance of the letter was found in the word, so that it can be returned to the function's caller. This flag should initially be set to `False`, and then set to `True` if the letter is found in the word.

b. The `play` function:

- To create a string of dashes, consider using the `*` operator.
- To find out if there are still dashes in `so_far`, consider using the `string find` function.
- Here is some high-level pseudocode describing one way you might write this function:

```
play(word)
   set so_far to be a string of dashes of the same length as word
   while so_far has dashes and num failed guesses is fewer than maximum allowed:
      get the user's guess
      use the replace_all function to find out if the guess was correct and
         to replace dashes in so_far if necessary
      display an appropriate message to the user (possibly drawing figure)
   if the maximum allowed failed guesses was exceeded:
      tell the user what the word was
   else
      congratulate the user on guessing the word correctly
```

c. The `main` function:

- Here's some high-level pseudocode for this function:

```
main()
   open the word.txt file
   read a word
   while the word is not the empty string and the user wants to continue:
      play the game with the word
      read a word
      if the word is not the empty string:
         see if the user wants to continue
```

d. The modification to the `play` function.

- Use a list to keep track of each guess. Each time you get a guess from the user, look to see whether the guess is already in the list. If it is not, put it in the list and do what your normally do.