HW4

Reading: Chapter 9

The goal of this assignment is to use `tkinter` to produce a class `hangman` that allows the user to play a simple game of hangman. It will loosely follow the strategy we used for the `Calculator` in class. Follow these guidelines:

1. Write a (module level, not in the class) function `mask` that takes two string arguments, the first a word the second containing exceptions. The function `mask` returns a string in which every character in the word has been replaced by '?' except for the characters in exceptions. For example:

```
>>> mask('APPLE','AE')
'A???E'
>>> mask('APPLE','')
'?????'
>>> mask('APPLE','PLEASE')
'APPLE'
>>>
```

2. The interface (see images below):
    a. `hangman` should inherit from Frame.
    b. Hangman Label & Entry displays the current state of the word
    c. Right Label & Entry – letters guessed correctly so far (no repeats listed)
    d. Wrong Label & Entry – letters guessed incorrectly so far (repeats not listed)
    e. 26 buttons, one for each letter of the alphabet

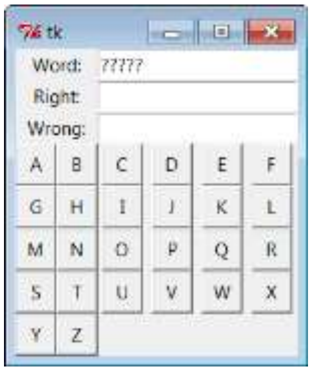
3. Gameplay. The player clicks on a letter button. Three things can happen:
    a. the letter is already in either the list of right or wrong letters. In this case nothing happens.
    b. the letter is in the word (but not in either list). Then that letter is added to the list of right letters, the word is remasked and the masked word (use `mask`) is displayed next to Hangman (in effect revealing the new letter). If the word is finished a `showinfo` box states "You Win!"
    c. the letter is not in the word (but not in either list). Then that letter is added to the list of wrong letters. If that list contains 6 or more letters, a showinfo box states "You Lose!"

4. Implementation details:
    a. `hangman` subclasses `Frame`

b. you may assume that the user will not edit the Entry's (this can be prevented but requires more than what we know now).
c. in addition to creating the interface, and calling Frame's `__init__`, `__init__` should take the given word, make it uppercase, and assign it to `self.word`
d. each button should set `command=cmd`, where `cmd` is a local function accepting one argument that defaults to the same letter on the button label (same thing we did with the calculator). `cmd` calls another method `click` to implement the gameplay above.

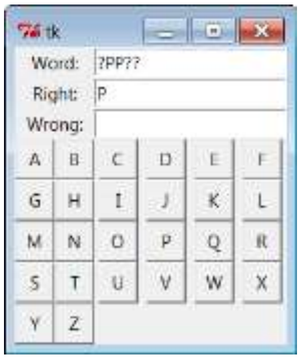
Here is an example of how the game works. First, start the game:

```
>>> root = Tk()
>>> hangman("APPLE",root).pack()  # will close when "X" cliced
```



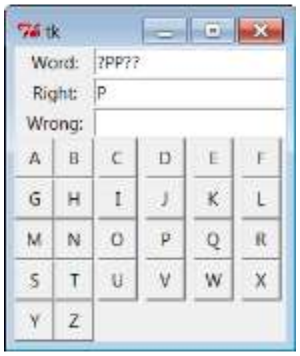
Word: ?????
Right:
Wrong:

click 'P'



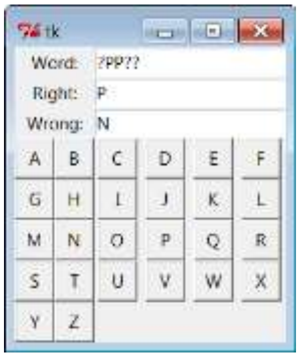
Word: ?PP??
Right: P
Wrong:

click 'P' again



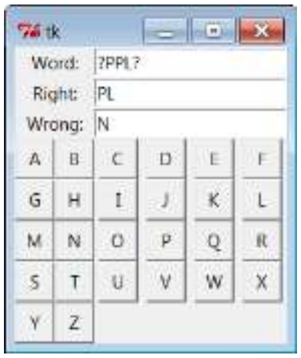
Word: ?PP??
Right: P
Wrong:

click 'N'



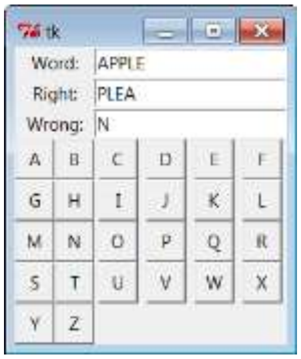
Word: ?PP??
Right: P
Wrong: N

'L'

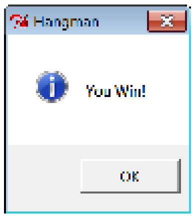


Word: ?PPL?
Right: PL
Wrong: N

‘E’,then ‘A’



Word: APPLE
Right: PLEA
Wrong: N



You Win!

could be ‘You Lose’ if there are 6 wrong letters