COS 125 Fall 2016 Lab #4

Due Friday October 14 EoD

This lab consists of modifying the "guess a number" game. It is intended as an exercise in incremental development. To submit your lab, create a document that has your source code along with some output demonstrating your games. Each question has two parts (function stubs and complete implementation) Show output for the second part only. You may submit not-quite-working programs for partial credit. Please use a monospaced font in your document. You can submit a text document or a pdf. Please put the following line at the top of the document

COS 125 Lab 4 [date] [Your Name]

1. The "guess a number" game is quite simplistic but can be turned into a more interesting game by using a wider range of numbers and adding winnings or bets to a repeating game. Consider that in general you can guess a number in no more than $\log_2(n)$ guesses, by using a "binary search" strategy where you always guess the midpoint of the range. For example, you should be able to guess a number between 1 and 128 in no more than 7 guesses because $\log_2(128) = 7$. (In case you get confused about logarithms this means that $2^7 = 128$). First guess 64; if the program says "too high", then you guess 32, etc. If the range is 1 – 100, then a player should be able to guess a number, on the average, in less than 7 guesses because $\log_2(100) \approx 6.644$.

Implement the following game a number in the range 1-100:

- The player starts with $0
- If the player fails to guess the number in 7 guesses, they lose $10
- If the player gets the number in 7 guesses, they win nothing
- Winnings are $7, $6, $5, $4, $3, $2 for getting the number in 1-6 guesses respectively.
- After each round the player is shown their total winnings (or losses) and asked if they want to play another round.
- Correct application of binary search should result in steady winnings, since 7 guesses is the worst case scenario.

1a. (30 points) Give some thought to how you might use functions to handle different parts of the game, and write the game logic with a set of function stubs as we have seen as the first step of incremental development. What to consider when developing function stubs:

- Consider the program action as a set of high-level steps where each step might be a function
- A function should handle one task (subproblem) only. For example "play game" is high level task, but involves several smaller tasks, such as "compute winnings or loss."
- If the same task is performed more than once with different variables, it is a good candidate for a function

1b. (30 points). Implement the game by adding logic and code to the function stubs.

NOTE: It is possible to implement the game in two ways: with global variables maintaining game state, where the variables are visible to the game functions; or with local variables only, where variables are passed as parameters to the functions. Either is a reasonable approach. Try to avoid mixing the two approaches

Extra Credit: 25 Points. Implement question 1b both ways.

2. Generalize the game from problem 1 as follows: The random number can range from 1 to 100,000. After generating a random number, determine which range it is in:

1-100	Allow 7 guesses
101-1000	Allow 10 guesses
1001-15000	Allow 14 guesses
15001–100000	Allow 17 guesses

If N is the maximum number of guesses allowed, the player loses $2 * N if they fail to guess in N guesses. The player wins or loses nothing if they take N guesses, and wins $N, $N-1, …,$2 for winning in 1 to N-1 guesses.

After generating a number at the start of each round, the range and the "betting" rules are displayed to the player, and they are asked if they want to proceed to play the round. After each round show the amount won or lost, as well as the total so far.

2a (20 points). Give some thought to how you might use functions to handle these changes and write function stubs as needed.

2b (20 points). Implement the game.

The source code from the slides is below.

```
import random
print "Hello! What is your name?"
name = raw_input()
print "Well, "+name+", I am thinking of a number between 1 and 20."
num = random.randint(1,20)
num_guesses = 0
guessed = False
while num_guesses < 6 and not guessed:
    print "Please enter a guess."
    guess = int(raw_input())
    num_guesses += 1
    if guess == num:
            guessed = True
    elif guess > num:
            print "Too High"
    else:
            print "Too Low"
if guessed:
    if num_guesses > 1:
        print "Congratulations! You got the number in", num_guesses, "guesses."
    else:
        print "Congratulations! You got the number in 1 guess."
else:
    print "You did not guess the number in",num_guesses,"guesses."
```