CPSC 1301, Computer Science I Programming Exercise 06

Week 06, Guess My Number Game

1 Introduction

This activity consists of three programming exercises. The following exercises are open book and open note. You are free to use any written documentation you wish. However, these are individual exercises, and you cannot consult with each other in writing your programs. Name your program `GuessMyNum_lastname.py`.

This programming exercise has three parts. The grade for each requirement is indicated, for a maximum of 100 points. At a minimum, your program must compile successfully and run. This exercise uses Python an infinite loop, if statements, and a random number generator. You do not have to use a list, and using a list is not recommended.

A starter template is shown below, but you do not have to use this. A sample output is also shown, and your exercise output must be similar to this output.

2 Exercise requirements

This exercise implements three functions implementing the bisection algorithm. The bisection method is an efficient way of finding a particular value in a sorted list. It takes a sorted list and a value, and finds the value in the list. First, it checks the "middle" element in the list. There are three possibilities: the value could match the "middle" element, the value could be higher than the 'middle" element of the list, or the value could be lower than the "middle" element. If the value matches, the function returns. If it's higher, the algorithm (recursively) calls itself with the top half of the list. If it's lower, the algorithm calls itself with the bottom half of the list.

For example, here is the output of a function call, the searched value being 7 and the sorted list being `[1 2 3 4 5 6 7 8 9 10]`.

```
>> int value = 7;
>> int[] list = {1,2,3,4,5,6,7,8,9,10};
>> bisection_search(value, list);
<< value is higher than 5
        //the ‘‘middle’’ value is 5.5, but we are doing integer division
        //the list is now set to {6,7,8,9,10}
>> bisection_search(value, list);
<< value is lower than 8
        //the middle value is 8
        //the list is now set to {6,7}
>> bisection_search(value, list);
<< value is higher than 6
        //the ‘‘middle’’ value is 6.5, but we are doing integer division
        //the list is now set to {7}
>> bisection_search(value, list);
<< value is equal to 7
        //the middle value is 7
        //the list is now set to {7}
<< the value searched for, 7, has been found
```

The bisection algorithm is nice because it is guaranteed to find an answer (or return if there is no answer) in logarithmic time of the size of the list. Only 10 repetitions of the function are necessary to find a result in a list of 1024 items, and only 20 repetitions to find a result in a list of 1,000,000 items. Mathematically, the time complexity of the bisection algorithm is $\log_2(n)$, where n is the length of the sorted list to be searched.

You are to implement three functions: `playGame()`, `play_the_computer()`, and `computer_plays_you()`.

playGame The function prints a greeting, prints the user choices, invites the user to make a choice, accepts input from the user, and calls the appropriate method depending on the user choice. See the sample output below.

play_the_computer The computer selects a random number between 1 and 100, and invites the human to guess the number. If the human's guess is too low, the computer tells the human that the guess is too low. If the human's guess is too high, the computer tells the human that the guess is too high. If the human's guess is correct, the computer tells the human how many guesses it took and returns to the main menu.

computer_plays_you The human selects a random number between 1 and 100, and invites the computer to guess the number. If the computer's guess is too low, the human tells the computer that the guess is too low. If the computer's guess is too high, the human tells the computer that the guess is too high. If the computer's guess is correct, the human tells the computer how many guesses it took and returns to the main menu.

random.randint Please check the documentation and make sure you know how to use this. Note that the starter template imports this method on line 7. The documentation states: " **random.randint(a, b)** Return a random integer N such that $a \leq N \leq b$."

system("cls") This just clears the screen and is strictly for cosmetic purposes. `system` is part of the `os` module. Read the documentation at `https://docs.python.org/3/library/os.html`.

3 Starter template

```
1  #!python
2  # Name: GuessMyNum.py
3  # Author: Your Name
4  # Date; current date
5  # Purpose: bisection algorithm, Guess My Number game
6
7  from os import system
8  from random import randint
9
10 def hello():
11     print("Hello from 'GuessMyNum.py'")
12
13 def play_the_computer():
14     system("cls")
15     print("Called play_the_computer()")
16     pass
17
18 def computer_plays_you():
19     system("cls")
20     print("Called computer_plays_you()")
21     pass
22
23 def playGame():
24     print("Welcome to the Guess My Number game")
25     while True:
26         pass
27
28 #main function executes the defined functions
```

```
29 if __name__ == '__main__':
30     hello()
31     playGame()
```

4 Sample output

```
Hello from 'GuessMyNum.py'
Welcome to the Guess My Number game

Enter 1 to play the computer.
Enter 2 to have the computer play you.
Enter 9 to exit.
What is your your choice: 1

Called play_the_computer()
Computer: I have chosen a number between 1 and 100
Computer: Please enter your guess
50
Computer: your guess was too high
Computer: Please enter your guess
25
Computer: your guess was too high
Computer: Please enter your guess
12
Computer: You guessed my number in 3 guesses

Enter 1 to play the computer.
Enter 2 to have the computer play you.
Enter 9 to exit.
What is your your choice: 2

Called computer_plays_you()
To Human: please choose a number between 1 and 100 and press <Enter> to continue.
Human: I have chosen a number between 1 and 100.
Please enter your guess:
Computer: My guess is 50
Human: enter 'low' if guess is too low, 'high' if guess is too high, and 'correct' if guess is correct:
high
Computer: My guess is 25
Human: enter 'low' if guess is too low, 'high' if guess is too high, and 'correct' if guess is correct:
high
Computer: My guess is 12
Human: enter 'low' if guess is too low, 'high' if guess is too high, and 'correct' if guess is correct:
low
Computer: My guess is 19
Human: enter 'low' if guess is too low, 'high' if guess is too high, and 'correct' if guess is correct:
correct
Human: You are correct. You guessed the number in 4 guesses

Enter 1 to play the computer.
Enter 2 to have the computer play you.
Enter 9 to exit.
What is your your choice: 3
```

```
I'm sorry, but I didn't understand.

Enter 1 to play the computer.
Enter 2 to have the computer play you.
Enter 9 to exit.
What is your your choice: 9

Thank you for playing. Goodbye
```