Computer Science 10A (Fall Term, 2021)
Introduction to Problem Solving in Python
Programming Assignment 7
DUE: Friday, Nov 19th, 11.59pm

Program Description:

This assignment focuses on `while` loops, random numbers, and lists.

Problem 1:

Your program allows the user to play a game in which the program thinks of a random integer and accepts guesses from the user until the user guesses the number correctly. After each incorrect guess, you will tell the user whether the correct answer is higher or lower. Your program must exactly reproduce the format and behavior of the logs in this document.

The log below shows one sample execution of your program. Your output will differ depending on the random numbers chosen and user input typed, but the overall output structure should match that shown below.

```
<< your intro message here >>

I'm thinking of a number between 1 and 100...
Your guess? 50
It's lower.
Your guess? 25
It's higher.
Your guess? 35
It's lower.
Your guess? 30
It's higher.
Your guess? 32
It's lower.
Your guess? 31
You got it right in 6 guesses!
Do you want to play again? y

I'm thinking of a number between 1 and 100...
Your guess? 50
It's higher.
Your guess? 75
It's lower.
Your guess? 65
It's lower.
Your guess? 64
You got it right in 4 guesses!
Do you want to play again? YES

I'm thinking of a number between 1 and 100...
Your guess? 60
It's lower.
Your guess? 20
It's higher.
Your guess? 30
It's higher.
Your guess? 40
It's higher.
Your guess? 50
It's lower.
```

```
Your guess? 47
It's higher.
Your guess? 49
You got it right in 7 guesses!
Do you want to play again? no

Overall results:
Total games   = 3
Total guesses = 17
Guesses/game  = 5.7
Best game     = 4
```

First, the program prints an introduction. Next, a series of guessing games is played. In each game, the computer chooses a random number between 1 and 100 inclusive. The game asks the user for guesses until the correct number is guessed. After each incorrect guess, the program gives a clue about whether the correct number is higher or lower than the guess. Once the user types the correct number, the game ends and the program reports how many guesses were needed.

After each game ends and the number of guesses is shown, the program asks the user if they would like to play again. Assume that the user will type a one-word string as the response to this question.

A new game should begin if the user's response starts with a lower- or upper-case Y. For example, answers such as "y", "Y", "yes", "YES", "Yes", or "yeehaw" all indicate that the user wants to play again. Any other response means that the user does not want to play again. For example, responses of "no", "No", "okay", "0", "certainly", and "hello" are all assumed to mean no.

Once the user chooses not to play again, the program prints overall statistics about all games. The total number of games, total guesses made in all games, average number of guesses per game (as a real number rounded to the nearest tenth), and best game (fewest guesses needed to solve any one game) are displayed.

Your statistics should be correct for any number of games or guesses ≥ 1. You may assume that no game will require one million or more guesses.

You should handle the special case where the user guesses the correct number on the first try. Print a message as follows:

```
I'm thinking of a number between 1 and 100...
Your guess? 71
You got it right in 1 guess!
```

Assume valid user input. When prompted for numbers, the user will type integers only, and they will be in proper ranges.

Implementation Guidelines:

```
<< your intro message here >>

I'm thinking of a number between 1 and 5...
Your guess? 2
It's higher.
Your guess? 4
It's lower.
Your guess? 3
You got it right in 3 guesses!
```

```
Do you want to play again? yes

I'm thinking of a number between 1 and 5...
Your guess? 3
It's higher.
Your guess? 5
You got it right in 2 guesses!
Do you want to play again? Nah

Overall results:
Total games   = 2
Total guesses = 5
Guesses/game  = 2.5
Best game     = 2
```

Define a **constant** for the maximum number used in the games. The previous page's log shows games from 1 to 100, but you should be able to change the constant value to use other ranges such as from 1 to 50 or any maximum.

Use your constant throughout your code and do not refer to the number 100 directly. Test your program by changing your constant and running it again to make sure that everything uses the new value. A guessing game for numbers from 1 to 5 would produce output such as that shown above.

Produce randomness using `randint`, as discussed in class.

Display rounded numbers using the built Python `round` function.

Read user yes/no answers using `input.` To test whether the user's response represents yes or no, use string functions discussed in lecture.

Produce repetition using `while` loops. You may also want to review fencepost loops and sentinel loops. Some students try to avoid properly using `while` loops by writing a function that calls itself, or a pair of functions A and B where A calls B and B calls A, creating a cycle of calls. Such solutions are not appropriate on this assignment and will result in a deduction.

Style Guidelines for Problem 1:

Structure your solution using functions that accept parameters and return values where appropriate. For full credit, you must have at least the following two functions other than `main` in your program:

1. a function to **play one game** with the user
   This function should *not* contain code to ask the user to play again. Nor should it play multiple games in one call.
2. a function to **report the overall statistics** to the user
   This function should print the statistics *only*, not do anything else such as `while` loops or playing games.

You may define more functions if you like. It is okay for some `print` statements to be in `main`, as long as you use good structure and `main` is a concise summary. For example, you can place the loop for multiple games and the prompt to play again in `main`.

Use whitespace and indentation properly. Give meaningful names to functions/variables and follow Python's naming standards. Localize variables. Put descriptive comments at the start of your program and each function. Since this program has longer functions, also put brief comments inside functions on complex sections of code.

Problem 2: Write a function called `is_sorted` that accepts a list of numbers as a parameter and return `True` if the list is sorted (nondecreasing) order and `False` otherwise. For example, if lists named `list1` and `list2` store `[16.1, 12.3, 22.2, 14.4]` and `[1.5, 4.3, 7.0, 19.5, 25.1, 46.2]`, respectively, the calls `is_sorted(list1)` and `is_sorted(list2)` should return `False` and `True`, respectively. Assume the list has at least one element. A one-element list is considered to be sorted.

Problem 3: Write a function called `collapse` that accepts a list of integers as a parameter and returns a new list containing the result of replacing each pair of integers with the sum of that pair. For example, is a list called `list1` stores the values `[7, 2, 8, 9, 4, 13, 7, 1, 9, 10]`, then the call of `collapse(list1)` should return a new list containing `[9, 17, 17, 8, 19]`. The first pair of the original list is collapsed into 9 (7+2), the second pair is collapsed into 17 (8+9), etc. If the list store an odd number of elements, the final elements is not collapsed. For example, if the list had been `[1, 2, 3, 4, 5]`, then the call would return `[3, 7, 5]`. Your function should not change the list that is passed as a parameter.

Guidelines:

Include a header comment at the beginning of your program with some basic information and a description of the program in your own words.

```
# Name Student
# COSI 10a, Fall 2021
# Programming Assignment #7
#
# Description: ...
```

You also need to include comments in your code.

For this assignment, you must limit yourself to the Python features covered up to lecture 19. Though we will cover more material while you work on this assignment, please do not use it on this assignment.

Submission and Grading:

All your python scripts should be inside a folder named `yourfirstname_ yourlastnamePA7`, then zip the folder into a zip file for submission. The zip file should have the following name: `yourfirstname_yourlastnamePA7.zip` (Please make sure to use exactly this file name, including identical capitalization).

Your program should be submitted via Latte before it is due (for late policy check the syllabus).

You will be graded on:

- **External Correctness:** The output of your program should match exactly what is in the question description. Programs that do not compile will not receive points for external correctness.
- **Internal Correctness**: Your source code should follow the stylistic guidelines shown in class. Remember to include the comment header at the beginning of your program and comment your code.