Problem Solving Team Members

If you are working digitally, record the name of each of your problem solving team members here.

Do not forget to ***add every team member's name***! Your instructor (or course assistant) may or may not use this to determine whether or not you participated in the problem solving session.

Nickolas Medina
Muhammad Yousaf Iqbal
Wendy Carrillo-Monarca

Problem 1

Consider a simple number guessing game during which the player is given 3 chances to guess a number between 1 and 10. If any guess that the player makes is correct, the game is over.

If they fail to guess the correct answer within 3 guesses, the answer is printed and the game is over.

You will implement a function that validates *one* of the player's guesses and ***returns*** a *string message* indicating whether or not it is out of range (less than 1 or greater than 10), too low, too high, or correct.

Assuming that you are using incremental development to write the function, what increments would you need to write to completely implement the entire game? List as many as you can think of.

Check with your instructor or a Course Assistant before moving to the next step.

I need to write the function guess that inputs an int saying “guess between 1 & 10” for the user. Add in if statements for when out of range. Add elif statements for either high or lower than 5 when the user inputs the answer and else statement to end if the user is correct.

```
def guess():
       guess= int(input”guess # between 1 & 10”)
       if guess < 1 or guess > 10:
              return “out of range”
       elif  guess < 5:
              return “too low”
       elif   guess >5:
              return “too high”
       else:
              return “correct ”
```

```
import guess

def test_guess_out_of_range():
    #setup
    guess = 0
    expected = “out of range”
    #invoke
    actual = guess.guess()
    #analyze
    assert actual == expected

def test_guess_out_of_range():
    #setup
    guess = 11
    expected = “out of range”
    #invoke
    actual = guess.guess()
    #analyze
    assert actual == expected
```

Problem 2

Begin implementing a function named "`check_guess`" that declares parameters for the `answer` and the player's `guess`.

Choose ***one*** of the increments that you and your team described in the previous problem and write ***only*** the minimum code necessary to implement that increment.

Next, write a pytest test function that verifies that your function is working properly. Remember, a good test has three parts:

- *setup* - create variables needed for input and to verify the correct results.
- *invoke* - call the function under test.
- *analyze* - verify that the function produced the correct results.

Problem 3

Repeat the steps of the previous problem with a new increment and unit test.

How will the function or the new unit test need to change compared to the first test to verify that the function returns the correct value?

Hint: if either the test or the function do not need to change at all, either your previous test was testing more than one thing, or you don't need this new test.

```
def guess():
        guess= int(input”guess # between 1 & 10”)
        if guess < 1 or guess > 10:
                return “out of range”
        elif  guess < 5:
                return “too low”
        elif   guess >5:
                return “too high”
        else:
                return “correct ”
```

```
import guess

def guess_too_low()
    #setup
    guess = 4
    expected = “too low”
    #invoke
    actual = guess.guess()
    #analyze
    assert actual == expected

def guess_too_high()
    #setup
    guess = 8
    expected = “too high”
    #invoke
    actual = guess.guess()
    #analyze
    assert actual == expected
```

```
import guess

def guess_is_correct()
    #setup
    guess = guess
    expected = “correct”
    #invoke
    actual = guess.guess()
    #analyze
```

```
def random_int():
            x =  randint(1,100)
           return x

    #The minimum guesses needed is 7
    if the #person is told whether the
    number is #higher or lower than
    their guess
```

Problem 4

Python's `random` module provides several functions that generate pseudorandom numbers within a given range, including:

- `random()` - returns a random floating point value in the range [0.0, 1.0]
- `randrange(a, b)` - returns a random integer value in the range [a, b).
- `randint(a, b)` - returns a random integer value in the range [a, b]

Using the random module, write the code necessary to generate a pseudorandom integer between 1 and 100 (inclusive).

What is the minimum number of guesses that a person would need to guarantee that they could guess the number?

Problem 5

Once your guess validating function is complete, you will be able to implement a `main` function that allows a player to play your game. Working together with your team, sketch out an algorithm using ***pseudocode*** to implement the guessing game. The player gets up to three guesses to guess the number, but the game ends if they guess correctly.

You should begin by generating a random number between 1 and 10 for the player to guess.

You should print feedback to the user after every guess.

```
def main():

correct=randint(1,10)
guess=int(input("Enter your guess: "))
counter=0
if counter=3
        break
elif guess <1 or guess>10:
        print("out of range")
        counter +1

elif guess>correct:
        print("Answer is lower")
        counter+1
elif guess<correct:
        print("Answer is higher")
        counter+1
else:
        print("You guessed correctly!")
        break

main()
```