Problem Solving Session

- The remainder of today's class will comprise the ***problem solving session*** (***PSS***).
- Your instructor will divide you into ***teams of 3 or 4 students***.
- Each team will ***work together*** to solve the following problems over the course of ***20-30 minutes***.
  - You may work on paper, a white board, or digitally as determined by your instructor.
  - You will submit your solution by pushing it to GitHub before the end of class.
- Your instructor will go over the solution before the end of class.
- If there is any time remaining, you will begin work on your homework assignment.

Class participation is a significant part of your grade (20%). This includes in class activities and the problem solving session.

Your graders will grade your participation by verifying that you pushed your solutions before the end of the class period each day.

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 TDD, what tests would you need to write to to verify that the function is working correctly? List as many as you can think of.

Remember: a test should be small (test *one* thing, not every kind of guess all at once) and fast.

test to see if it higher than the answer
test to see if it lower
out of range

```
def test_out_of_range():
    answer = 11
    guess = 2
    assert check_guess(guess, answer) == 'Guess out of range'
```

```
def check_guess(guess, answer):
    return “Guess out of range”
```

Problem 2

Choose *one* of your test cases from the previous problem and write the code to implement the test case as a function that will work with pytest.

Remember, at this point you are testing a function that does not exist. What will you name a function that validates the player's guess? What parameters do you think that it will need?

Then write your implementation of the function with the *minimum code* required to make your test pass.

Problem 3

Repeat the steps of the previous problem with a new test case.

How will the function under test need to change to make the new test pass?

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

```
def test_check_range_correct():
    guess = 0
    answer = 0
    assert check_guess(guess, answer) ==
“correct”
```

```
def check_guess(guess,answer):
    if(guess == answer):
        return “Correct”
    else:
        return “Guess out of range”
```

```
import random
def check_guess(guess,answer):
    if(guess == answer):
        return “Correct”
    elif(guess > answer):
        return “Too high”
    elif(guess < answer):
        return “Too low”
    else:
         return “Guess out of range”

def main():
     answer= random.randint(1, 10)
     i = 0
     while(i<3):
          i +=1
          guess = int(input("Enter a number between 1 - 10: "))
          print(check_guess(guess, answer))
          if(i == 3):
               g = 'Game over'
               print(g)
               return
```

Problem 4

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.

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.