Create Task 2a-2d

2a.
I made QUIZlet 2.0, a word memorizer tool, using Python 3.6.2. This program’s purpose is to help users memorize terms and their definitions more effectively. The program fulfills this purpose by first letting users enter in term-definition pairs. The program then lets the user study those pairs until they are ready for a quiz. Then, the program quizzes the user about those terms, and finishes up by printing information about how the user did on the quiz, as well as the terms the user messed up on. This video illustrates the above features by showing how the program first asks the user to enter in terms and their definitions and allows the user “study” until they are ready, then quizzes the user, then finally prints out how many terms the user answered correctly as well as terms which the user messed up on.

2b.
An incremental process was when I drew a flowchart detailing the flow of the program, including how the program has three functions collect_dict(), study(), quiz(), that collects term-definition-pairs, helps user study, and quizzes the user, respectively, and a fourth function run_quiz() that calls the three former functions in the correct order. Another incremental process incorporated into the code is a for loop that prints blank lines until the counter reaches 100. An iterative process was when I decided to improve the program by using the lower() method to make it non-case-sensitive for users. Both these processes were done independently. A difficulty encountered was when my tester found that the program always calculates the user’s quiz-scores 1-digit higher than it actually is. I resolved this by assigning points_scored to 0. Another difficulty was encountered when my tester noticed that the quiz feature was printed directly below the “study” feature and users taking the quiz can cheat off the answers displayed in the study section. I resolved this by printing 300 blank lines between the “study” and “quiz” portions. These 2 difficulties were resolved by collaborating with a classmate.
An **algorithm** in my program is a **for-loop** that **includes 3 sets of if-else statements**. The **for-loop** iterates over each item in the dictionary and **asks** the **user** to enter the term which corresponds to a given **definition**. The **1st set** of if-else-statements functions by **adding 1** each to user’s **score** and **streak-point** if the user’s **answer** = key of the **dictionary-item**. Else, it resets streak to 0. This demonstrates use of **logical concepts** since if-statements use Booleans. The **2nd set** of if-else-statements determine which motivational-message to print based on their **streak-points**. If the streak-point >= 2, it runs the algorithm’s **3rd set** of if-else-statements, which uses **modulus**—a **mathematical concept**—to determine which motivational message to display by using modulus 2 to determine the parity of the streak-score. Additionally, if streak-point < 2, the algorithm prints out a message that tells the user to try harder. All in all, this **entire algorithm helps to achieve** intended purpose of the program by quizzing the **user**, and the 3 smaller algorithms (the if-else-statements) included within the algorithm determines the score and streak-points of the user and uses that information to print adequate motivational messages. This entire algorithm’s **developed independently**.

```python
for (a, b) in users_dict.items():
    guess = str(input("Input the term that has the definition of " + b + ": "))
    if guess.lower() == a.lower():
        points_scored += 1
        streak += 1
    else:
        words_missed.append(a)
        streak = 0
    if streak >= 2:
        if streak % 2 == 1:
            print("You’re on fire!! YAY...
")
        else:
            print("You’re doing great! Keep it up!
")
    elif streak == 1:
        print("Keep it up!! You’re almost there...
")
    else:
        print("Oh come on, I know that you can do it...
")
```
An abstraction developed individually is quiz(), which is used in the function run_quiz(). quiz() quizzes the user, records terms the user messes up on, calculates total points_scored and streak-points, and gives the user motivational messages based on their streak-points. Example of using a mathematical concept is using modulus-2 to determine the parity of the user’s streak-points. Example of a logical concept is an if-else statement that determines if the user answered correctly. My function does this by returning a Boolean value (True or False) which describes whether the user’s answer equals correct answer, and using that value to determine whether or not to run add 1 each to points_scored and streak-points. Having this abstraction helped manage the complexity of this program by one, improving readability, and two, making debugging easier. Firstly, since all the code regarding the quiz is in a function, running the quiz itself only requires quiz(). Second, this abstraction made debugging significantly easier. An example of this is when I found out that the quiz was not functioning the way I had expected it to, so I immediately started looking for errors in the quiz() function rather than having to look through my entire program for the error.

```python
def quiz():
    for y in range(300):
        print()
    points_scored = 0
    words_missed = []
    print("QUIZ TIME!!!")
    streak = 0
    for (a, b) in users_dict.items():
        guess = str(input("Input the term that has the definition of " + b + ": "))
        if guess.lower() == a.lower():
            points_scored += 1
            streak += 1
        else:
            words_missed.append(a)
            streak = 0
        if streak >= 2:
            if streak % 2 == 1:
                print("You're on fire!! YAYY...
            else:
                print("You're doing great! Keep it up!!")
        elif streak == 1:
            print("Keep it up!! You're almost there...")
        else:
            print("Oh come on, I know that you can do it...")
    print("\nGame over. You got", points_scored, "out of", num_terms, "!!!")
    return words_missed
```