A program which help kids solve arthmetic calculations including: addition substraction and multiplication,

Each category should include 10 question each with an increasing diffculity FOR each category,

Each question have a time limit ranging from 10 to 15 depending on the difficulity of the question

IF the user fails to enter the correct answer IN the giving time he loses a point otherwise he earn a point

FOR each difficulity the numbers ranges from:

1 -> 1,10

2 -> 10,20

3 -> 20,30

4 -> 10,40

5 -> 20,50

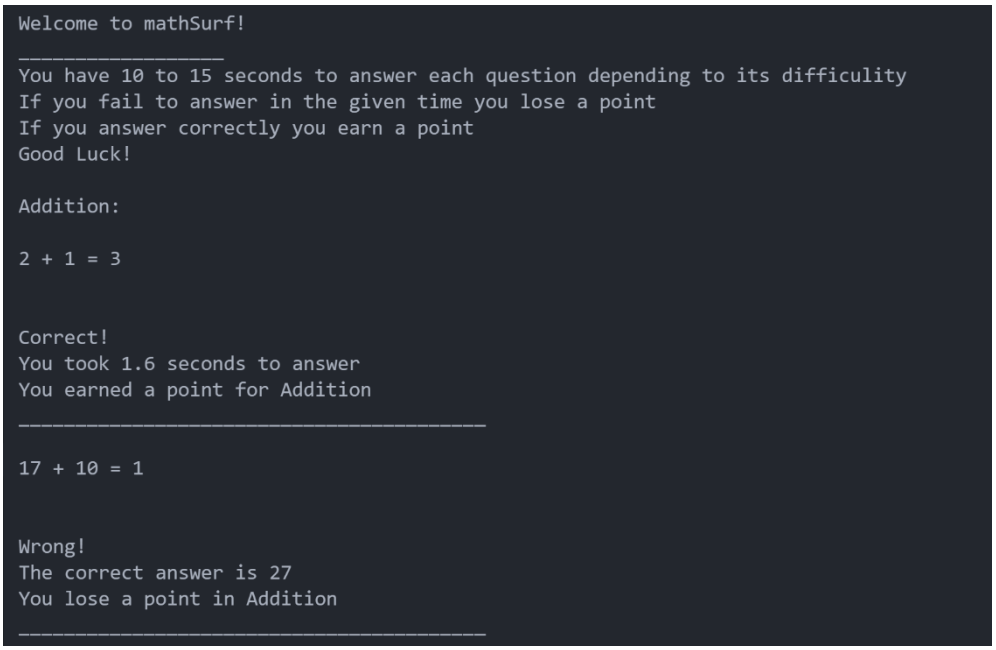
6 -> 30,60

7 -> 40,70

8 -> 50,80

9 -> 60,90

10-> 70,100



DEFINE THE PROGRAM CONSTANTS (CATEGORIES, DIFFICULITIES …)

DEFINE FUNCTION generate_randInt_based_on_diffuclity_and_category(difficulity, category, categories):

"""

Generate a random integer based on the difficulity and category

(int) -> int

"""

DEFINE FUNCTION swap_to_get_biggest_number(num1, num2):

"""

Swap the numbers to get the biggest number

(int, int) -> int

"""

IF num1 > num2:

RETURN num1,num2

ELSE:

RETURN num2, num1

```
DEFINE FUNCTION get_two_num_based_on_difficulty_and_category(difficulty, category , categories):
    """
    Get two random numbers based on the difficulty
    (int) -> int, int
    """
```

```
DEFINE FUNCTION insert_question_with_answer(category, difficulty, categories):
    """
    Insert a question and its answer based on the category and difficulty to the questions dictionary
    (str, int) -> str
    """
    IF category EQUALS "Addition":
            GET two random number based on DIFFICULITY
            INSERT A QUESTION WITH ITS ACTUAL ANSWER
    ELSEIF category EQUALS "Substraction":
        GET two random number based on DIFFICULITY
         INSERT A QUESTION WITH ITS ACTUAL ANSWER
    ELSEIF category EQUALS "Multiplication":
        GET two random number based on DIFFICULITY
        INSERT A QUESTION WITH ITS ACTUAL ANSWER
```

DEFINE FUNCTION generate_questions_and_answers(number_of_questions, categories):

"""

Generate questions and answers based on the number of questions

(int) -> None

"""

INITILIZE ALL THE DICTIONARIED BASE ON THE CATEGORIES

DEFINE FUNCTION readInt(message):

"""

Read a valid integer from the user

(str) -> int

"""

```
DEFINE FUNCTION validate_answer(user_answer, question, start_time, end_time, category):
    """
    Validate the answer and update the points and time_allocated
    (int, int, float, float, str) -> None
    """
    GET TIME_TOOK
    GET TIME_LIMIT
    IF user_answer EQUALS actual_answer:
        OUTPUT("Correct!")
        OUTPUT(f"You took {time_took} seconds to answer")
        IF time_took <= time_limit:
            points[category] += 1
            OUTPUT(f"You earned a point FOR {category}")
        ELSE:
            points[category] -= 1
            OUTPUT(f"You lost a point FOR {category}")
    ELSE:
        OUTPUT("Wrong!")
        OUTPUT(f"The correct answer is {actual_answer}")
        points[category] -= 1
        OUTPUT(f"You lose a point IN {category}")
    OUTPUT("_____\n")
```

```
DEFINE FUNCTION ask_questions(categories):
    """
    Ask the questions from the questions dictionary FOR each category
    (str) -> None
    """
```

```
DEFINE FUNCTION OUTPUT_result(categories):
    """
    Print the result
    """
    OUTPUT(categories_line)
    OUTPUT(points_line)
    OUTPUT(time_line)
    OUTPUT(AveragePoint)
    OUTPUT(AverageTime)
    IF average_points >= SuccessRate:
        OUTPUT("Congrats, You passed the test!")
    ELSE:
        OUTPUT("Sorry, you did not pass, consider taking the test again")
```

```
DEFINE FUNCTION main():
    """
    Main entry point for the program
    """
    CALL initialize_questions_dict(Categories)
    CALL initialize_time_dict(Categories)
    CALL initialize_points_dict(Categories)
    CALL generate_questions_and_answers(Num_of_questions_for_each_category, Categories)
    OUTPUT("Welcome to mathSurf!")
    OUTPUT("_____")
    OUTPUT("You have 10 to 15 seconds to answer each question depending to its difficulity")
    OUTPUT("If you fail to answer IN the given time you lose a point")
    OUTPUT("If you answer correctly you earn a point")
    OUTPUT("Good Luck!")

    WAIT 5 SECONDS

    CALL ask_questions(Categories)
    CALL OUTPUT_result(Categories)

_____

IF __name__ EQUALS "__main__":
    CALL main()

_____
```