|  |  |  |  |
| --- | --- | --- | --- |
| NAME: | Sarcol, Joshua S | DATE: | 09/27/2023 |

PRACTICE EXERCISE # 5.1

# LE 5.11 Average of Even Numbers and Product of Odd Numbers

Program that will ask 10 integers and display the average of all even numbers and the product of all odd numbers entered by the user

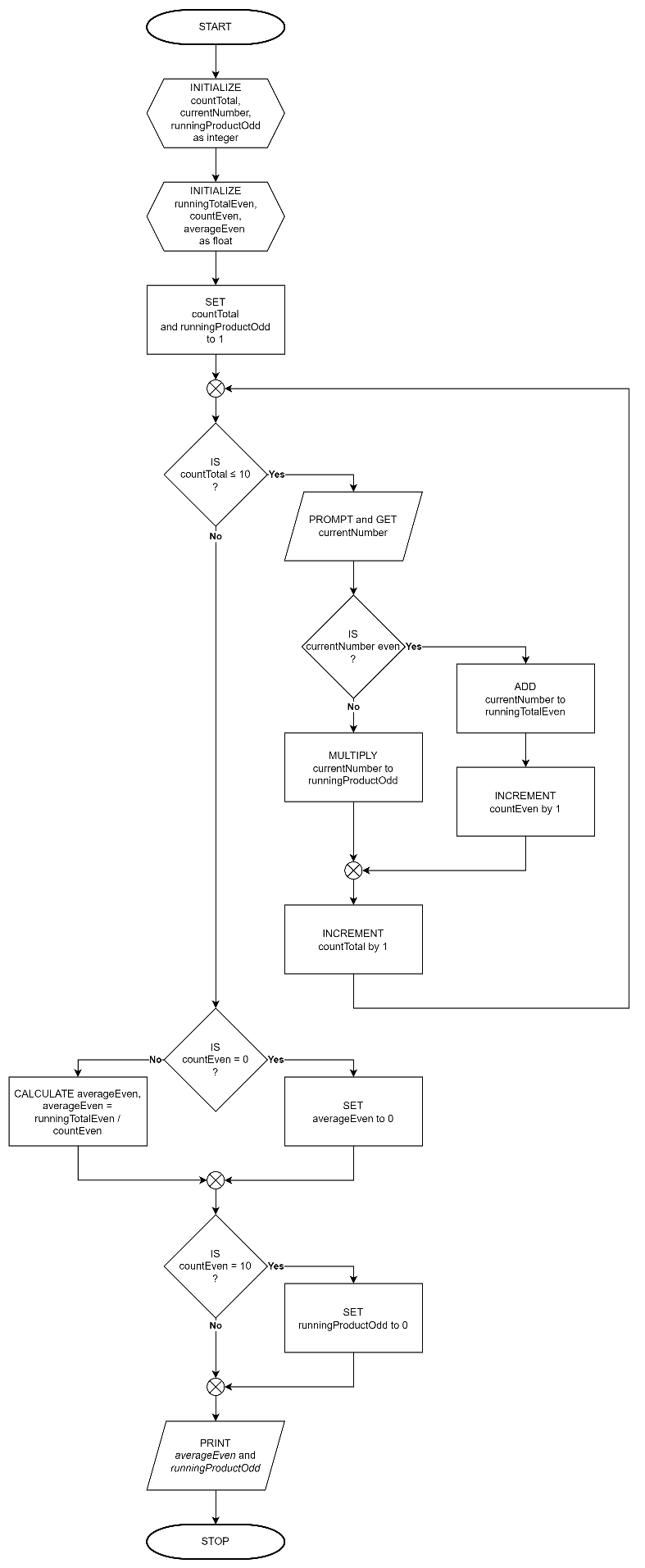
## Pseudocode: Average of Even Numbers and Product of Odd Numbers

START

1. INITIALIZE countTotal, currentNumber, runningProductOdd as integer
2. INITIALIZE runningTotalEven, countEven, averageEven as float
3. SET countTotal and runningProductOdd to 1
4. DO
   1. PROMPT and GET currentNumber
   2. IF currentNumber is even
      1. ADD currentNumber to runningTotalEven
      2. INCREMENT countEven by 1
   3. ELSE
      1. MULTIPLY currentNumber to runningProductOdd
   4. ENDIF
   5. INCREMENT countTotal by 1
5. WHILE countTotal ≤ 10
6. IF countEven is 0
   1. SET averageEven to 0
7. ELSE
   1. CALCULATE averageEven, averageEven = runningTotalEven / countEven
8. ENDIF
9. IF countEven is 10
   1. SET runningProductOdd to 0
10. ENDIF
11. PRINT *averageEven* and *runningProductOdd*

STOP

## Flowchart: Average of Even Numbers and Product of Odd Numbers



# LE 5.12 Passcode Lock

Program that will ask the correct numeric passcode before the user can continue his task. The user is given only three trials to enter the correct passcode. When the user hit the correct passcode the program will display “ACCESS GRANTED”, otherwise it will display “ACCESS DENIED” after three attempts.

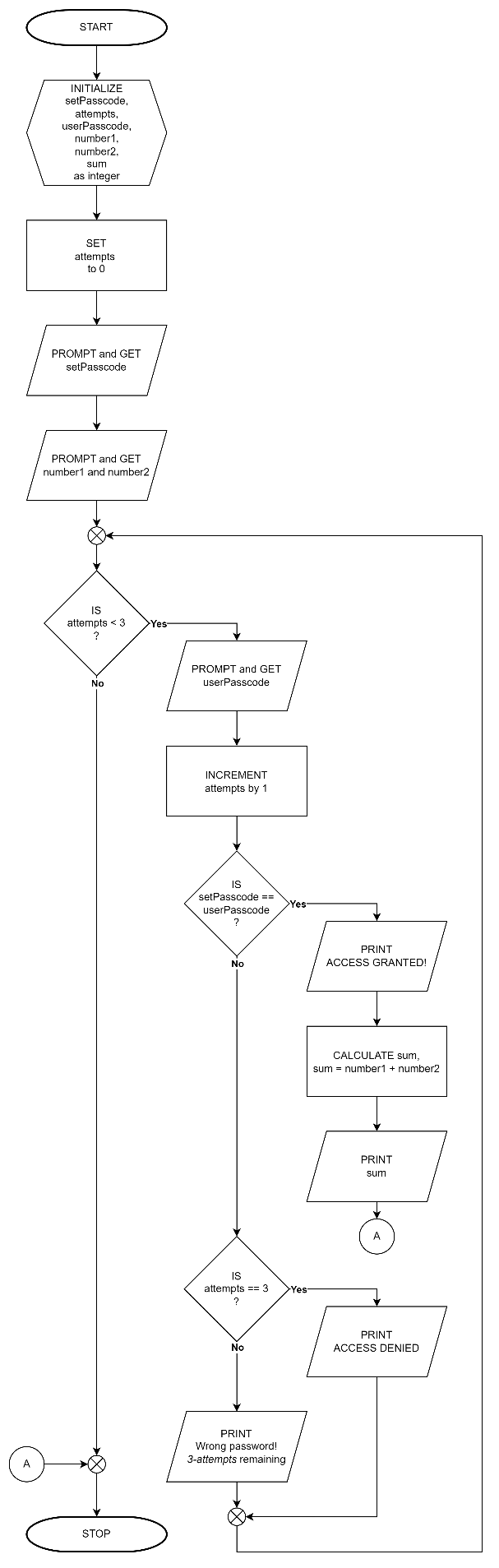
## Pseudocode: Passcode Lock

START

1. INITIALIZE setPasscode, attempts, userPasscode, number1, number2, sum as integer
2. SET attempts to 0
3. PROMPT and GET setPasscode
4. PROMPT and GET number1 and number2
5. DO
   1. PROMPT and GET userPasscode
   2. INCREMENT attempts by 1
   3. IF setPasscode == userPasscode
      1. PRINT ACCESS GRANTED!
      2. CALCULATE sum, sum = number1 + number2
      3. PRINT sum
      4. BREAK from loop
   4. ELSE IF attempts == 3
      1. PRINT ACCESS DENIED
   5. ELSE
      1. PRINT Wrong password! *3-attempts* remaining
   6. ENDIF
6. WHILE attempts < 3

STOP

## Flowchart: Passcode Lock



# LE 5.13 Problem Solver Menu

Create a program that lets the user choose an operation (power problem solver, factorial problem solver, or finding roots for quadratic equations using quadratic formula) from the menu. The program will always go back to the menu and lets the user choose again an operation until the user would like to quit using the app.

Note: For quadratic problems, it still follow the rules below:

If both a and b are zero, there is no solution

If a is zero, there is only one root (-c/b).

If the discriminate (b2-4ac) is negative, there are no real roots

For all other combinations, there are two roots.

## Pseudocode: Problem Solver Menu

START

1. INITIALIZE userInput as character
2. DO
   1. DO
      1. PROMPT [p]ower, [f]actorial, [r]oots of quadratic equation, [q]uit
      2. GET userInput
   2. WHILE userInput is not one of ‘p’, ‘f’, ‘r’, or ‘q’
   3. INITIALIZE base, power, factorial, solution as integer
   4. SET solution to 1
   5. SWITCH userInput
      1. CASE ‘p’
         1. PROMPT and GET base and power
         2. CALCULATE solution, solution = base^power
         3. PRINT solution
         4. BREAK from case
      2. CASE ‘f’
         1. PROMPT and GET factorial
         2. INITIALIZE and SET i to 1
         3. FOR (int i = 1; i <= factorial; i++)
            1. MULTIPLY i to solution
         4. ENDFOR
         5. PRINT solution
         6. BREAK from case
      3. CASE ‘r’
         1. quadraticSolver()
         2. BREAK from case
   6. ENDCASE
3. WHILE userInput is not ‘q’

STOP

quadraticSolver()

1. INITIALIZE , b, c, discriminant, x1, x2 as float
2. PROMPT and GET a, b, and c
3. CALCULATE discriminant, discriminant = b^2 - 4\*a\*c
4. IF a == b and b == 0
   1. PRINT No solutions found.
5. ELSE IF a == 0
   1. CALCULATE x1, x1 = -1\*c/b
   2. PRINT x1
6. ELSE IF discriminant < 0
   1. PRINT No real-numbered solutions found.
7. ELSE
   1. CALCULATE x1, x1 = (-1 \* b + sqrt(discriminant)) / (2 \* a)
   2. CALCULATE x2, x2 = (-1 \* b - sqrt(discriminant)) / (2 \* a);
   3. IF x1 == x2
      1. PRINT x1
   4. ELSE
      1. PRINT x1, x2
   5. ENDIF
8. ENDIF

RETURN

## Flowchart: Problem Solver Menu

