NAME: Enrique Miguel O. Arcenas, Christian Andrie G. Asne **DATE:** Nov 14, 2023

AlgoExercise 6.13

LE6_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.

main()

```
START
  1. ASSIGN variable choice, base, power, num, a, b, c, answerPowerSolver,
     answerFactorialSolver,answerQuadraticSolver
  2. DO
     a. CALL displayMenu()
     b. GET int, choice
     2.1 IF choice == 1
         a. PROMPT and GET base, base
         b. PROMPT and GET power, power
         c. CALL answerPowerSolver = powerSolver(base, power)
         d. DISPLAY answerPowerSolver
     2.2 ELSE IF choice == 2
         a. PROMPT and GET integer, num
         b. CALL answerFactorialSolver = factorialSolver(num)
         c. DISPLAY answerFactorialSolver
     2.3 ELSE IF choice == 3
         a. PROMPT and GET a, a
         b. PROMPT and GET b, b
         d. PROMPT and GET c, c
         e. CALL answerQuadraticSolver = quadraticSolver(a, b, c)
            2.3.1 IF answerQuadraticSolver == 1
                a. DISPLAY No Solution
            2.3.2 ELSE IF answerQuadraticSolver == 2
                a. DISPLAY firstRoot
            2.3.3 ELSE IF answerQuadraticSolver == 3
                a. DISPLAY No real roots
            2.3.4 ELSE IF answerQuadraticSolver == 4
                a. DISPLAY firstroot and secondroot
            2.3.4 ENDIF
        2.4 ENDIF
    3.  WHILE choice != 4
    4.  ENDDO
END
```

displayMenu()

```
START
    1.  DISPLAY Power Problem Solver
    2.  DISPLAY Factorial Problem Solver
    3.  DISPLAY Quadratic Equation Root Problem Solver
    4.  DISPLAY Quit
    5.  DISPLAY Enter number:
END
```

powerSolver(int base, int p)

```
START
    1.  INITIALIZE answerPower = 1, i
    2.  FOR i = 1; i is less or equal to p; increment i
        2.1 CALCULATE answerPower*=base
    3.  ENDFOR
RETURN answerPower
```

factorialSolver(int num)

```
START
    1.  INITIALIZE factorial = 1
    2.  WHILE num > 0
        2.1 CALCULATE factorial*=num
        2.2 DECREMENT num
    3.  ENDWHILE
RETURN factorial
```

quadraticSolver(int a, int b, int c,)

```
START
    1. INITIALIZE discriminant = (b*b)-(4*a*c), flag
    2. IF INPUT a and b is 0
        2.1 flag = 1
    3. ELSE IF INPUT a is 0
        3.1 flag = 2
    4. ELSE IF Discriminant is negative
        4.1 flag = 3
    5. ELSE for all other combination
        5.1 CALCULATE the first root, firstRoot = ((-b+sqrt(discriminant))/(2*a))
        5.2 CALCULATE the second root, secondRoot = ((-b-sqrt(discriminant))/(2*a))
        5.3 flag = 4
    6. ENDIF
RETURN flag
```