|  |  |  |  |
| --- | --- | --- | --- |
| NAME: | Albert Justin Joe Enabe, Joshua Sarcol | DATE: | 11/13/2023 |

PRACTICE EXERCISE # 6.1

# LE6\_11 Add Digits

Create a program that extracts and adds the two least significant digits of an integer.

Function Prototypes:

*int extractFirstLSD (int num);*

*int extractSecondLSD (int num);*

*int sumDigits (int x, int y);*

## Pseudocode: LE6\_11 Add Digits

main()

START

1. INITIALIZE originalNumber as integer
2. PROMPT and GET originalNumber
3. CALL module, onesPlace = extractFirstLSD(originalNumber)
4. CALL module, tensPlace = extractSecondLSD(originalNumber)
5. CALL module, sum = sumDigits(onesPlace, tensPlace)
6. PRINT sum

STOP

extractFirstLSD(num)

START

1. CALCULATE result, result = num % 10

RETURN result

extractSecondLSD(num)

START

1. CALCULATE result, result = (num / 10) % 10

RETURN result

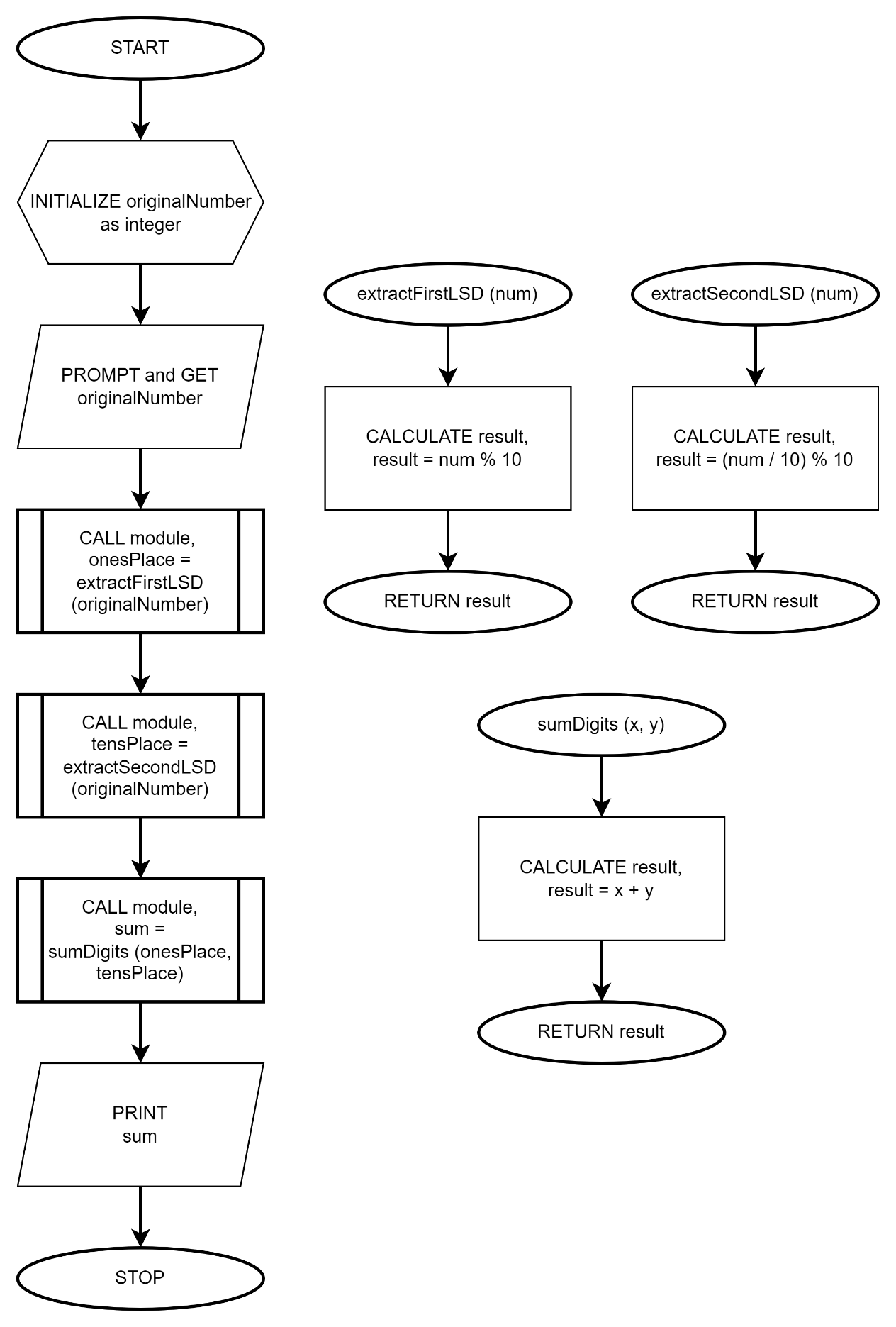
sumDigits(x, y)

START

1. CALCULATE result, result = x + y

RETURN result

## Flowchart: LE6\_11 Add Digits



# LE6\_12 Vowel and Consonant

Determine whether the letter entered by the user is vowel or consonant. Do not use predefine functions (isAlpha, tolower, toupper, etc.).

Function Prototypes:

bool isLetter (char ch);

bool isVowel (char ch);

## Pseudocode: LE6\_12 Vowel and Consonant

main()

START

1. INITIALIZE letter as character
2. PROMPT and GET letter
3. CALL module, resultLetter = isLetter(letter)
4. IF !resultLetter
   1. PRINT INVALID INPUT!
5. ELSE
   1. CALL module, resultVowel = isVowel(letter)
   2. IF resultVowel
      1. PRINT VOWEL!
   3. ELSE
      1. PRINT CONSONANT!
   4. ENDIF
6. ENDIF

STOP

isLetter(ch)

START

1. IF ch is a letter
   1. SET result to TRUE
2. ELSE
   1. SET result to FALSE
3. ENDIF

RETURN result

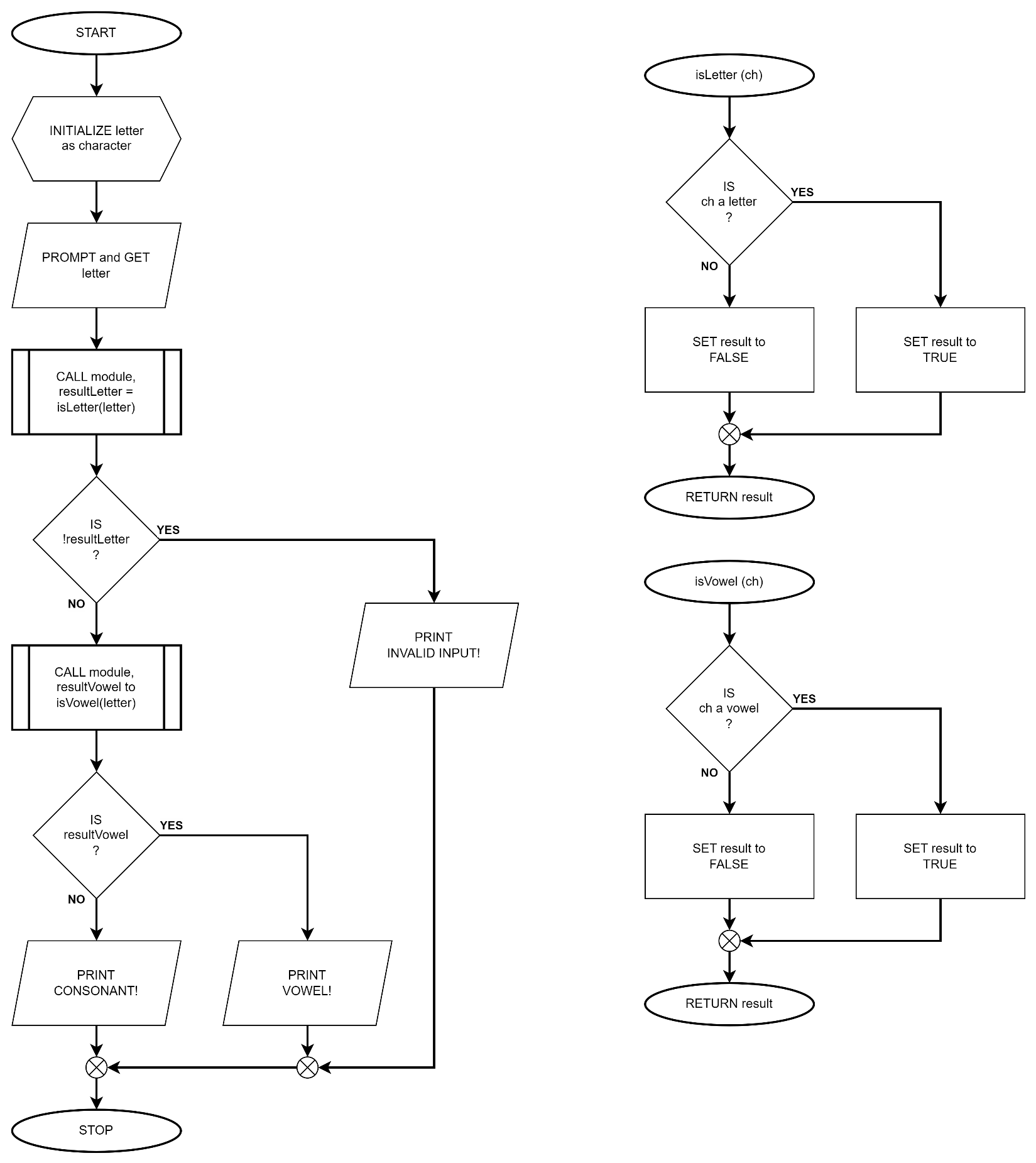
isVowel(ch)

START

1. IF ch is a vowel
   1. SET result to TRUE
2. ELSE
   1. SET result to FALSE
3. ENDIF

RETURN result

## Flowchart: LE6\_12 Vowel and Consonant



# 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.

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.

Function Prototypes:

*void displayMenu ();*

*int powerSolver (int base, int p);*

*int factorialSolver (int num);*

*int quadraticSolver (int a, int b, int c); ===> you may set two root variables as global*

## Pseudocode: LE6\_13 Problem Solver Menu

main()

START

1. INITIALIZE userInput as character
2. INITIALIZE x1, x2 as global float
3. DO
   1. DO
      1. CALL module, displayMenu()
      2. GET userInput
   2. WHILE userInput is not one of ‘p’, ‘f’, ‘r’, or ‘q’
   3. INITIALIZE a, b, c, 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. CALL module, solution = powerSolver(base, power)
         3. PRINT solution
         4. BREAK from case
      2. CASE ‘f’
         1. PROMPT and GET factorial
         2. CALL module, solution = factorialSolver(factorial)
         3. PRINT solution
         4. BREAK from case
      3. CASE ‘r’
         1. PROMPT and GET a, b, and c
         2. CALL module, solution = quadraticSolver(a, b, c)
         3. SWITCH solution
            1. CASE 0

PRINT No solutions found.

BREAK from case

* + - * 1. CASE 1

PRINT x1

BREAK from case

* + - * 1. CASE 2

PRINT x1 and x2

BREAK from case

* + - * 1. ENDCASE
      1. BREAK from case
  1. ENDCASE

1. WHILE userInput is not ‘q’

STOP

displayMenu()

START

1. PRINT [p]ower, [f]actorial, [r]oots of quadratic equation, [q]uit

RETURN

powerSolver(base, p)

START

1. CALCULATE result, result = base^p

RETURN result

factorialSolver(num)

START

1. INITIALIZE and SET i to 1
2. FOR (int i = 1; i <= num; i++)
   1. MULTIPLY i to result
3. ENDFOR

RETURN result

quadraticSolver(a, b, c)

START

1. INITIALIZE discriminant as float
2. CALCULATE discriminant, discriminant = b^2 - 4\*a\*c
3. IF a == 0 and b == 0
   1. SET result to 0
4. ELSE IF a == 0
   1. CALCULATE x1, x1 = -1\*c/b
   2. SET result to 1
5. ELSE IF discriminant < 0
   1. SET result to 0
6. 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. SET result to 1
   4. ELSE
      1. SET result to 2
   5. ENDIF
7. ENDIF

RETURN result

## Flowchart: LE6\_13 Problem Solver Menu

