9/25/2023

Requirements Exercise

Christopher Upham, Sophia Herrell

ETSU
SE1

Requirements

Mandatory

- Program must explain rules to user (F)
- Must prompt user for input (F)
- Must verify user input (NF) //ask kinser
- Must take user input (F)
  - 2 numbers
- Must provide descriptive directions for prompts
- Must validate user input
  - Must check for overflow errors
  - Must make sure numbers > 0
- Must add numbers together (F)
- Must save sum (F)
- Must take the least significant digit of the sum and add it to the remaining digits (F)
- Must repeat this process until it produces a single digit number (F)
  - Must complete this process recursively (NF)
- Must use variables in the place of literal constants (NF)

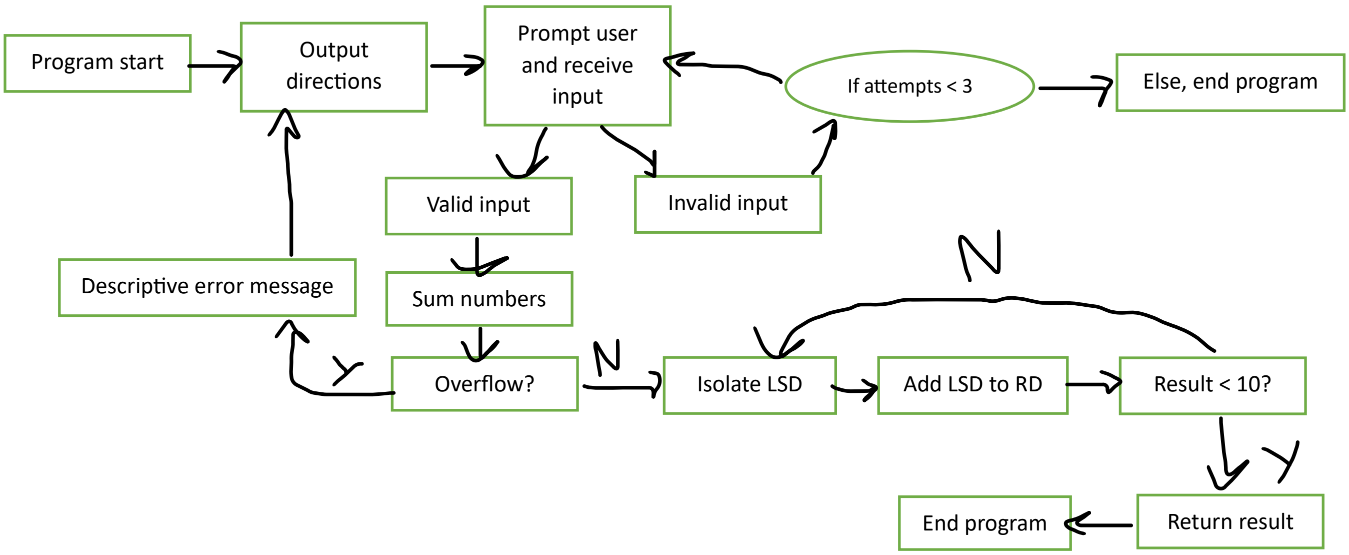
Non-mandatory

- 2 numbers will be positive integers only (F)
- Solution can be implemented in C# (NF)
  - User can interact with program via command line (F)
- If user enters invalid input, program should output an error message and re-prompt
- Player should not be able to enter invalid input more than three times (F)
- Code should be thoroughly commented (NF)
- Program can output final result to user upon completion of recursive method
- Program can have recursion counter

Design

Note: LSD = least significant digit (rightmost), RD = remaining digits

Example: sum = 853; LSD = 3, RD = 85



- Program start
    - When you execute program through Visual Studio
- Output directions
    - Using console.writeline, display detailed directions for the user
- Prompt user for input using console.writeline to prompt and console.readline to receive
- Validate user input
    - Allow user to enter incorrect input up to three times
    - Input must be a nonnegative integer
    - Must not result in a stack overflow
    - If user entered valid input, proceed
    - Otherwise, end program after 3 attempts
- Sum numbers together
    - If sum results in stack overflow, gracefully handle exception by prompting the user again for smaller inputs
    - If no other exceptions occur, proceed
- Enter recursive method
    - Isolate least significant digit (LSD)
        - If number is already a single digit, skip this step
        - Implement this using modular arithmetic and storing the result of and dividing the number by 10 to remove last digit
    - Add LSD obtained in previous step to remaining number
    - Repeat above process recursively until the result is a single digit number
    - Return single digit number
    - Return number of recursions (optional)

- Output final result
- Output goodbye message
- End program

Coded solution in progress. Requirements also subject to modification. `