CYB 220 Homework #3 - Advanced Calculator

Due: Friday Oct 18, 2024, 11:59 pm on CANVAS.
Turn in: your code and screenshots (at least two, one for functional testing and one for security testing).
Points: 100 points

Objective: Practice defensive programming, especially input validation and check for integer overflow.

In this assignment, you will develop an advanced calculator.

- There are five arithmetic operations: +, -, *, / and % (remainder).
- (20 pts) Two operands will be accepted: (1) two integers or (2) one integer and one string (up to 100 characters, contains letters a-z and/or A-Z only). Some sample user input can be: 2+4, 2*ABc, GOOD-2, 311/45, AAAAAAAAAAAA*5.
    - If two strings are entered as operands (eg. Abc+def, xyz-mno), reject the input. Only accept two integers or one integer and one string.
    - If the integer is > INT_MAX or <0, reject it
    - If the string has more than 100 characters, reject the input
- User will be asked to enter everything in one line, the two operands and the arithmetic operator (No whitespace allowed!), such as abc*3, 34-5, 2+abcdefg, 1*abc (The operator is entered between the two operands). Users are only allowed to enter uppercase and lowercase letters, digits (0-9), and arithmetic operators (+, -, *, /, %). If other characters are given, reject the input and ask for a new one.
- (20 pts) If the two operands are integers, then simply do the arithmetic operations (addition, subtraction, multiplication, division, and remainder). Don't forget to check for potential integer overflow/underflow and divide by zero.
- (40 pts) If the two operands contain an integer and a string (eg. Acb+3, 7*kkkk, 4-homework).
    - Only four operators are allowed: +, -, *, /. (if % is entered, reject the input)
    - + operator, "shift right" a letter by a number. Eg. abc+1 = bcd, abc+2 = cde, abc+23=xyz, abc+24=yza, abc+25=zab, abc+26=abc, abc+27=bcd, abc+89=lmn, ABC+1=BCD, ABC+24=YZA, ABC+25=ZAB.
    - – operator, "shift left" a letter by a number. Eg. abc-1=zab, abc-3=xyz, ABC-5=VWX, ABC-22=EFG.
    - * operator, expand/repeat the string n (the integer operand) times, up to 1024 characters. Eg, abcd*2=abcdabcd, mnmnmn*3=mnmnmnmnmnmnmnmnmn. If after the expansion, the string is longer than 1024 characters (abc*500), print a message about it ("result string is too long" or something similar) and only keep 1024 characters and ignore the later characters.
    - / operator, cut the string from the end by n (the integer operand) characters. Eg. abcde/2=abc, homework/3=homew, homework/7=h, homework/8="" (empty string). If the integer operand is larger than the string length, just leave an empty string as the result, eg homework/10="" (empty string).
    - % operator, reject the input.

Note:

- Make your calculator run multiple times until the user enters "exit" to terminate the program.
- For all of the numbers, use **signed int** as their data type (The goal of this homework is to practice checking for integer overflow and input validation, I don't care of getting 5/2=2 ☺).

- The user input will be in one line, so you need to parse it to get the two operands and the operator, keep in mind you may get abc*3 or 4+abc as input.
- Assume you only get positive numbers as inputs. So you don't need to consider abc*-2 , 3/-200, -10+3.
- Input validation
    - User input numbers should be in the range of an int variable (>=0 and <=INT_MAX).
    - User input strings should be up to 100 characters.
    - User input should only contain digits (0-9), letters (A-Z, a-z), and the five operators (+, - ,*, /, %).
    - String Operand contains letters a-z and A-Z only.
- Check for integer errors
    - Check for integer overflow (SaftInt class, built-in functions, or other methods)
    - Divide by 0 and modulo (%) 0 are not allowed.

Testing:

- Your program should provide meaning messages when rejecting a user input. Such as, number is too big, input includes bad characters, the resulting string is longer than 1024 or etc.
- Test your program with whatever input you want to use. But the following specific input are required to be shown on the screenshots.
- (8 pts) Functional testing: Test your program to make sure it performs as expected (especially the operations on strings). Please use the exact invalid inputs below and take screenshot(s) for functional testing.
    - ABCabc+30
    - 220+cyb
    - CYBERsecurity-5
    - 6*zzzz
    - CYB*220
    - Cybersecurity/8
    - 20/Cybersecurity
    - 100%7
- (12 pts) Security testing: Test your program with invalid inputs below (such as out-of-range numbers, longer strings, input contains not-allowed characters, and etc), to ensure your program can handle them as expected. Please use exact invalid inputs below, and take a screenshot(s) for security testing.
    - 2000000000+300000000
    - 123+4000000000
    - 77777*10000
    - -acb123
    - abcde*500,
    - 9+abc;def
    - 123+5+6+7
    - Abc-xyz
    - 3*aaaaa…aaaaa(more than 100 'a's)
    - 500%0 (not allowed)
    - Abc%4 (should reject)
    - 3 + 0 (white space in the input)