Design and Testing for Question 1

Design Description

The program is designed to stimulate a calculator for subtracting the second input big integer from the first big integer and display the result on screen. The program allows only positive integers and each up to 50 digits as input.

Design Rationale

The structure of the program include three main sections, namely, store, subtract, display result.

Store:

The designer of the program decided to store these two big integers in two char arrays with the size of 50. The elements of two char arrays will all be initialised to '0'. Then an integer will be read in one digit a time. Before reading into the array, the program will set a filter to catch and remove any potential wrong input. Once the number has been read in, the program will use the function to store each digit reversely in order to make the subtraction easier starting from the lower digit place to higher digit place. The designer has learned how to reverse a string from internet and use the logic to implement a function to store numbers into char arrays in a reverse order.

Subtract:

After that, based on the length of two input numbers, the designer has consider three scenarios of subtraction.

First, when the size of first number array is bigger than the second number array, the subtraction is simply num1 minus num2 from low digit place moving up to higher digit place.

Second, when the size of first number array is smaller than the second number, then the subtraction becomes - (num2 minus num1).

Third, when two numbers have the same length, i.e. with same number of digit, the program needs to find out which number is actually bigger. That's when the designer decided to use the function strcmp from cstring library to compare the values of two char arrays. That's also the reason why the designer uses char array to store two numbers instead of converting into int arrays straightway once the numbers are read in.

A same rule applies to all these three scenarios that is if at one digit place, the value of num1 digit is smaller than num2 digit. Then the value of num1 at the current digit place will carry/add 10 and reduce one from its higher digit place as the way we do math in the primary school.

Display the result:

After the subtraction, the program will remove any potential zero in front of the actual digits of the result. Then converting the result into int array and printing out the result in a reverse order from the highest digit place to the lowest.

Testing Strategy

The testing strategy the designer applied is to check if the system will give the results the designer expected in three different scenarios by inputting various data sets. The testing strategy also takes potential input error handling into consideration. So at the testing stage, the designer deliberately enters some wrong format input such as a letter, a symbol, a negative number or a number with more than 50 digits. By deliberately making these kind of mistakes, the designer can test how the program copes with these to ensure it works robustly. Additionally, the designer has tested the results for each block of codes (e.g. reversing an array) to ensure it works before putting each section of codes together as a completed program.

Test Results

- **Testing wrong input**
  - Entering a letter
    - E.g. I entered 'a', 'abc', 'letter', the program all displayed: "Invalid input, try again!"
  - Entering a symbol
    - E.g. I entered '!', '£', '^%£&', the program displayed: "Invalid input, try again!"
  - Entering a number more than 50 digit
    - E.g. I entered '11111111111111111111111111111111111111111111111111111111111111 11111111111111111111111111111111111111111111111111111111111111 111111', the program also displayed: "Invalid input, try again!"

The system allows the user keeps trying input numbers until they enter the right format number with less than 50 digits.

- **Scenario 1 – num1 is bigger than num2 as num1 has a longer length (has more digits).**
  - Num1 – 22222 num2 – 111 result: 22111
  - Num1 – 1234567 num2 – 9999 result: 1224568
  - Num1 - 239834095803945862440835983452184985298358
    Num2 – 939542309853120721934217021372984729812
    Result: 238894553494092741718901766430812000568546

- **Scenario 2 – num1 is smaller than num2 as num1 has a shorter length (has fewer digits).**
  - Num1 – 111 num2 –22222 result: -22111
  - Num1 – 9999 num2 – 1234567 result: -1224568
  - Num1 – 939542309853120721934217021372984729812
    Num2 – 239834095803945862440835983452184985298358
    Result: -238894553494092741718901766430812000568546

- **Scenario 3 – num1 and num2 have the same length.**
  - Num1 > Num2

- Num1 – 654321 num2 –543210 result: 111111

- Num1< Num2
  - Num1 – 123456 num2 – 234567 result: -111111
- Num1 = num2
  - Num1 – 239834095803945862440835983452184985298358
    Num2 – 239834095803945862440835983452184985298358
    Result: 0