CSCI 140 -- Project 2 (Big Integers)

Due Thursday, 04/11/2019 (TTh Section)
Due Wednesday, 04/10/2019 (MW Section)

If you need to work with very large integers such as applications in data encryption and decryption, you may not be able to use data type ***int*** or even the new ***long long int***. A string or an array of integers can be used to store a large, but it is easier to perform basic arithmetic operations with an array of integers. For example, the integer 1234 could be stored in an ***int*** array ***a*** by setting a[0] to 1, a[1] to 2, a[2] to 3, and a[3] to 4 and the number of digits (4) can be stored in a separate variable. However, it is much easier to perform arithmetic operations when digits are stored starting from the end of the array by placing 4 in a[19], 3 in a[18], 2 in a[17], and 1 in a[16] (assume an array of 20 elements). Therefore, you **MUST** store digits in that order for this project. Unused digits will be initialized to 0 and you might need a variable such as `numDigits` to keep track of the number of actual digits.

Here is the representation for 1234 + 72 = 1306 when the maximum number of digits is 20:

Index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	2	3	4

`numDigits: 4`

Index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	7	2

`numDigits: 2`

Index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	3	0	6

`numDigits: 4`

Write a program to perform addition and subtraction operations with large integers up to 20 digits in length. Your program shall be able to input a positive integer, an operator (+ or -), and another positive integer (assume at least one space between the two items). It then outputs the sum or difference of the two numbers if the operation is legal. For subtraction, you can assume that the first operand is larger than or equal to the second operand unless extra credit option b is attempted. Your program will perform the calculation by implementing the usual paper-and-pencil addition or subtraction algorithm (adding or subtracting one digit at a time with carrying or borrowing as needed). The result is stored in an ***int*** array and it is then printed to the screen.

Your program must allow the user to continue until the expression 0 % 0 is entered. The following is a sample run of the program and you must follow its user-interface.

```
[Some information/instructions including your name]

Enter an expression --> 1234 + 72<Enter>
1234 + 72 = 1306

Enter an expression --> 1234 - 72<Enter>
1234 - 72 = 1162

Enter an expression --> 123456789012345678901234 - 1<Enter>
Invalid operand (too large).

Enter an expression --> 123456789 - 123A5<Enter>
Invalid operand (bad digit).

Enter an expression --> 77777777777777777777 + 22222222222222222222<Enter>
77777777777777777777 + 22222222222222222222 = 99999999999999999999

Enter an expression --> 99999999999999999999 + 1<Enter>
Integer overflow.

Enter an expression --> 0 % 0<Enter>
There are 3 valid operations.
Thanks for using my program.  Good bye!
```

Your program shall reject an invalid operator, an operand with invalid character, or an operand that has a length longer than maximum allowed digits. If the result is an integer with more than the maximum number of digits (i.e., more than 20 digits), then your program shall issue a message saying it has encountered "Integer overflow." You should be able to change the maximum length of the integers by changing only one globally defined named constant. It is probably easier to check out your program with shorter-length integers first. Make sure to properly utilize functions in your program or points will be deducted. Here are the **four required functions**, but feel free to add more functions as applicable to your design: convert an operand as string to an ***int*** array, add (two big integers as two ***int*** arrays), subtract (two big integers as two ***int*** arrays), and output one big integer (one ***int*** array).

Hint: It is easiest to input each operand as a C++ string, but you can also read each digit as value of type ***char***. Do not attempt to read a number as type ***int*** because it can be larger than an ***int*** can hold. After they are read into the program, each character should be changed to a value of type ***int*** and should be stored in one slot of an ***int*** array (convert a character like '1' to integer value 1).

Extra credit: You can earn up to 4 additional points for implementing one the following features (do not have to implement all features). You can submit one version of your program including extra credit option, but you might want to start with a regular version and save a copy of that version in case there are problems with the extra credit. Clearly specify which extra credit feature in your write up.

a. Allow multiplication (i.e, `1234 * 72`).
b. Allow division (i.e, `1234 / 72`).
c. Allow positive or negative integers (i.e, `-1234 + 72` or `72 - 1234`).

Please provide documentation and applying good coding style because it is part of the grade. Do not forget to use the provided template. You must run the specified test cases in the sample input/output first and able to pass provided test cases in a separate file as well. Please submit a hardcopy of the following items **in a folder** if flash drive is included (can also submit source code ahead of time via Canvas and a folder is not needed). **You can preferably submit just a hardcopy of the title page or this project sheet and all remaining items electronically via Canvas (a PDF file of all printouts and actual .cpp file)**.

1. Title page with name, class, project number, and relevant information about your program (compiler and system used, file names).
2. Notes about your program (status of your program at the minimum, but you should discuss issues/bugs regarding your program). Indicate extra credit option if applicable.
3. Pseudocode (separate pseudocode for each function).
4. Printouts of any input/output (run above test cases as the minimum and add additional test cases as needed).
5. A printout of the source code.
6. A copy of your source code on a flash drive or Canvas -- source code (.cpp) for this program.

Your program will be graded as follow:

- Correctness/Efficiency: 25 points
- Test Cases: 5 points
- Pseudocode: 5 points
- Documentation/Coding Style: 5 points
