CPSC 1160 SECTION 001 INSTRUCTOR: GLADYS MONAGAN

ASSIGNMENT #2: C++ ARRAYS

DUE DATE WITH BRIGHTSPACE: WEDNESDAY, SEPTEMBER 21, 2022 AT 11:50 PM

THE LEARNING OUTCOMES

- write functions that have parameters that are passed-by-value and passed-by-reference
- add two big integers
- input and output the big integers
- user (static) arrays: you need to pass the size of the array as a parameter
- test user input for validity in a program that interacts with the user
- unit test the functions with the lightweight unit testing framework doctest (doctest tutorial)

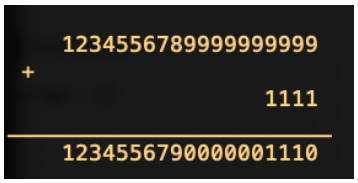
READINGS

Read §2.8, §4.6, §5.8, §6.12, §6.13, §6.14, §6.15 and chapter 7 of the textbook.

PROBLEM AT HAND

Write a program that interacts with the user

- reads two big integers (up to and including MAX_NUM_DIGITS)
- determines if the input integers are valid (i.e. they consist of digits exclusively):
  the big integers are positive integers or zero without a unary + symbol
- adds the two big integers is possible
  - outputs both the input integers and the '+' symbol in a single column
    i.e. the least significant digits to the most significant digits align
    with a plus and a line to indicate that below is the total
    Example



    prints out an error message if the addition is not possible
- continue additions until the user wants to stop

BONUS

Ask the user for the base and do all the computations in that base.
Put in a README.txt that you attempted the bonus.

IMPLEMENTATION

- For this assignment you must use statically allocated arrays, i.e. use arrays which have their size determined at compile time. Do not use dynamic arrays for this assignment, do not use variable length arrays because ISO C++ does not allow them, and do not use the STL vector.
- Use an array of int to store the number i.e. each element of the array will store a single digit.
  Do all your calculations in base 10 (unless you do the bonus)
  You need to keep track of the number of digits that are used in the array. Do not include "leading zeros" in the array of ints nor in the output.
  How are you going to store the number zero if you are not to store "leading zeros"?
- The arrays that store the big integers should be allocated to have at most MAX_NUM_DIGITS. Set MAX_NUM_DIGITS to 72. If the sum cannot be stored in MAX_NUM_DIGITS, report this as an error.
- Use the functions defined in bigInts.cpp and test these functions in bigInts_unittest.cpp
- Write a program to interact with the user called calc_bigInts.cpp

In calc_bigInts.cpp, deal with input errors gracefully, giving the user some feedback about the error.
Do not just "exit" the program if there is an error.
Ask the user if the user wants to continue. As in Assignment #1, your program should be well structured and should not have breaks and continues, etc.

TO SUBMIT WITH BRIGHTSPACE AS A SINGLE ZIP FILE:

1. A file called **bitInts.cpp** that contains the function prototypes needed. Read the documentation and directives. Add more functions as needed.
   Brightspace has the 'starting' file **bitInts.cpp**

2. A file called **calc_bigInts.cpp** that implements the program that interacts with the user.
   In that file **calc_bigInts** there is an (ugly!!) #include bigInts.cpp.
   Document the program.
   Brightspace has the 'starting' file **calc_bigInts.cpp**

3. Some "screen captures" or "print screens" that show your program above **cal_bigInts** works.
   Put the screen captures in a MS Word document or a pdf.

4. A file called **bigInts_unittest.cpp** that contains the unit tests and has and (ugly!) #include bigInts.cpp. You only need to write *more* tests for the two functions
   fromStringtoArray and add
   The documentation in this file will come from the names of the TEST_CASEs
   Brightspace has the 'starting' file **bigInts_unittestcpp**

5. The file doctest.h
6. Possible a README.txt: you must include in the README.txt that you have done the bonus