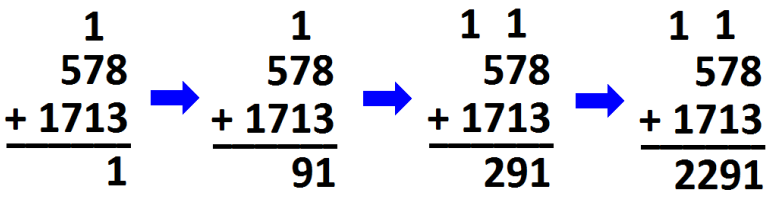
Lab Assignment #7

This week's assignment will have you write a program that performs addition of very large numbers via arrays. Also, this assignment will give you practice creating a multiple-file program whose parts can be compiled separately. Feel free to work in pairs and ask for help early.

You may recall performing long addition by hand:



You add the digits in the ones column, carrying the one if the sum is ten or more. Then moving to the tens column and repeating the process until you're done.

We have discussed how integers have a maximum value they can represent, which isn't very large (2,147,483,647). We can represent a much larger number in an array through, if each digit is stored in one of the array's elements. As you can see on the left below, the numbers may have different lengths, and that can make it difficult to align the columns for ones, tens, etc. But, as you see on the right, if we reverse the digits in the numbers, then the ones column is at index 0, the tens index 1, etc. This is visually backwards, but we are now able to perform long addition using the matching columns.

Your program will consist of three files:

- A header file named `bigint.h`
- An implementation file for the header file, named `bigint.cpp`
- A main program file named `main.cpp`

The main program, `main.cpp`, has been provided for you. You will use this file as-is, except for the addition of your preamble.

The `bigint.h` header file must contain the prototypes of the three functions described next. You are provided the prologues for the three functions, which you use to implement the behavior of each. Within the prologues you will find the function name, a description of what it does, pre- and post-conditions,

and the return value. Also, you'll find each parameter explained, along with whether it acts as an input or input and output together:

```
/** addBigIntegers(num1, num1_size, num2, num2_size, result_num)
 *  @desc     adds the number represented by the arrays using long addition,
 *            placing the result into result_array
 *  @param    int num1[]        [in/out] - a number stored as an array
 *  @param    int num1_size     [in]     - the # of digits in the first number
 *  @param    int num2[]        [in/out] - a second number stored as an array
 *  @param    int num2_size     [in]     - the # of digits in the second number
 *  @param    int result_num[]  [in/out] - an array to store the result number
 *  @pre      declared size of result_num is > max(num1_size, num2_size)
 *  @post     the number representing the sum of num1 and num2 is stored in
 *            the result_num array
 *  @post     the digits in num1, num2 and result_num are in proper order
 *  @return   the number of digits in result_num array
**/

/** printBigInteger(outputstream, num, size)
 *  @desc     prints the number represented by the digits in the num array
 *            to the output stream
 *  @param    ostream & outputstream [in/out] - the stream to write to
 *  @param    int num[]              [in/out] - the number to print as an array
 *  @param    int size               [in]     - the size of the array
 *  @pre      outputstream is open for writing
 *  @post     the digits in num have been written to the output stream
 *  @return   none
**/

/** reverseArray(array, size)
 *  @desc     reverses the elements in an array
 *  @param    int array[] [in/out] - the array to reverse the elements of
 *  @param    int size    [in]     - the used size of the array
 *  @pre      none
 *  @post     the first element is now the last and the last is now the first;
 *            the second element is now the second to last...
 *  @return   none
**/
```

The file `bigint.cpp` must contain the definitions for the function prototypes in the header file.

It is your task to create the header and implementation files and then write the correct function prototypes and definitions to make the main work.

I recommend you create the header file first, adding the prototypes. Then, work on the implementation file. I strongly urge you to work out the algorithm needed for the `addBigIntegers` function, before you try to write any of the code.

Submitting your work

Each of your source code files must have a preamble at the top (see the programming rules and guidelines document in Blackboard for more on the preamble).

Make sure you are in the folder with your source code files (use the `ls` command) and then run the following to create a zip archive of them:

```
$ mkdir lastname_firstname_lab07
$ cp main.cpp lastname_firstname_lab07/main.cpp
$ cp bigint.h lastname_firstname_lab07/bigint.h
$ cp bigint.cpp lastname_firstname_lab07/bigint.cpp
$ zip -r lastname_firstname_lab07.zip lastname_firstname_lab07/
```

You'll need to change `lastname` and `firstname` to your actual last and first names (**use lowercase letters**) in the steps above. Once you have your zipfile, you can use that file as your submission for the assignment in Blackboard.

If you are working in pairs, then you should have both of your names included as comments in the source code file's preamble (see the programming rules and guidelines document in Blackboard for more on the preamble). Also, write both your names to the notes section in the submission form when submitting to Blackboard.