CPSC 1160 SECTION 001 INSTRUCTOR: GLADYS MONAGAN

ASSIGNMENT #7: DYNAMIC MEMORY

DUE DATE WITH BRIGHTSPACE: NOVEMBER 2 AT 11:50 PM

THE LEARNING OUTCOMES

- to work with dynamic memory including writing code without memory leaks
- to manipulate a dynamic data structure (a C++ struct) to add, subtract, and multiply two big integers
- to revisit Assignment #2 but to use a different data structure
- unit test the functions with the lightweight unit testing framework doctest (doctest tutorial)

READINGS

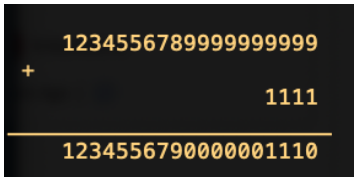
chapter 11 Pointers and Dynamic Memory

- §11.11, §11.12, §11.14 and §11.15 will be covered later in CPSC 1160

PROBLEM AT HAND

Finish the program that interacts with the user

- Reads one big integer, reads an operator (as a character), and reads a second big integer
- 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, subtracts, multiplies the two big integers if 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 calculations 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.

DATA STRUCTURE

We represent a 'big integer using the following C++ structure

```
struct BigInt {
   int numDigits;
   int *digits; // array of coefficients from
                // least significant digit to most significant digit

};
```

You will only be representing non-negative numbers.

Do your calculations in base 10 (the default argument) and for bonus marks make it any base up to hexadecimal.

We suggest that you program `number, character, readBigInt` and `printBigInt` first. You can read the number as a string and then reverse the characters, convert them to digits, and store them in the `digits` of the struct `BigInt`. Since you can get the length of a string, you can find out how big the size of `digits` needs to be: except that you need to remove leading zeros.
Think of how you are going to represent the constant `0` and write the function that creates that constant in `zeroConstant`

If after the addition/subtraction/multiplication, the resulting big number has leading zeros, you need to allocate a new array of the right (and smaller) size and copy the coefficients from the old array into the new array. Do not forget to deallocate the array that is no longer needed. Do not just "patch" `printBigInt` to handle the extra 0s.

Every big number must always have the numDigits correspond to the number of elements in digits without leading zeros (possibly except for the constant 0).

In your main program, read the big numbers, read the operator, do the computation outputting the answer and continuing until the user wants to stop.

TO SUBMIT WITH BRIGHTSPACE AS A SINGLE ZIP FILE:

1. A file called **BigInts.h**
   Fill in all the functions required and add other functions if you want.
2. A file called **BigInts.cpp**
   Implement the functions required and add other functions if you want.
3. An application file called **testBigInts.cpp**
   We'll work on it in the lab.
4. The **Makefile**
5. A file called **unittestBigInts.cpp**
   We'll work on it in the lab.
6. The file doctest.h
7. Possible a README.txt: you must include in the README.txt that you have done the bonus

The final version of this assignment will be given after the lab.