Project A5

Due Date: April 30

Purpose

Once again, this project will allow you to a) implement an algorithm for a problem we talked about in class, and b) bolster your C++ skills by writing more object-oriented code.

Problem

There are many applications that require the use of very large integers. Some examples include encryption algorithms in cryptography, problems in astronomy (the universe is rather large), and blockchain implementation. But C++ can only handle **signed** integers in the range of $[-2^{n-1}, 2^{n-1} - 1]$, where $n = 64$ bits. This is $[-9223372036854775808, 9223372036854775807]$. This means that integers can be up to only 19 digits long, very insufficient for the kinds of problems listed above. Your job is to create a class in C++ called `bigNum` that will handle arbitrarily large integers. Functionality includes being able to input large integers, and carry out addition, subtraction, and multiplication operations.

Input

The program should allow the user to type in a large integer or create one through a constructor. Input will be through the usual `cin` statement. For a little practicality, we will limit the number of digits to 32. The integer can be preceded by a minus sign to indicate a negative value (but not a plus sign).

Output

Output will be handled by the usual `cout` statement and will display whatever large integer the user desires. A sample `main()` looks like this:

```
int main() {
    bigNum a("12345678901234567890");
    bigNum b, c;

    cout << "Enter a value: ";
    cin >> b;

    c = a * b;

    cout << "Result: " << c << endl;
    return 0;
}
```

A sample run is shown below:

```
Enter a value: 2
Result: 24691357802469135780
```

Specifics

- The program should be able to multiply and add two large integers using the `*` and `+` operators. Note that one (or both) of the integers may be negative, so that the addition might in fact be subtraction. The result should have the appropriate sign.
- As usual, follow good OOP practices. As before, there is some starting code that you **must** follow. This is on the course web page and is shown below:

```
// bigNums.cpp
// Starting code for A5

#include <iostream>
#include <string>

using namespace std;

enum signVals {POS, NEG};

class bigNum {
private:
    string value;
    signVals sign;

    // Any private method prototypes here

public:
    // Default constructor
    bigNum() {value = "0"; sign = POS;}

    // Parameterized constructor
    bigNum(string s);

    // Overloaded * operator
    bigNum operator*(/* parameter(s) here */) const;

    // Overloaded + operator
    bigNum operator+(/* parameter(s) here */) const;

    // Input operator prototype here

    // Output operator prototype here
};
```

  You must use this code as written, but you can add any additional code as described in the comments.

- You **do not** have to implement Karatsuba multiplication! Just do a brute-force kind of addition/multiplication. The program only handles integers up to 32 digits, so efficiency is not a concern.
- You do not have to do any error checking of input numbers.
- Remember to adequately comment your classes, methods, and all variables.

Notes

- AI gave me a solution that did not work correctly. It also contained some questionable programming practices. Just sayin'.
- Be sure to adequately test your program.
- Zip your code together using the same naming convention as before, as in `gousieA5.zip`. **Zip only your code! I do not want either of the** `__MACOSX` **or** `cmake-build-debug` **folders!** Figure out how to zip up only your `.cpp` and `.h` files, not the entire CLion (or other) folder! Turn in your zip file in Canvas before the due date.
- A printed version of your source code is due in the envelope on my office door on May 1^{st}. This printout should contain text in black on a white background. Write/print and sign the Wheaton Honor Code Pledge on what you turn in: "I have abided by the Wheaton College Honor Code in this work."

An Algorithm for the Organization of Information.
– The vague title of the paper by Adelson-Velskii and Landis describing AVL trees.