Programming Project #2
EGRE246 Spring 2018
Big Integers

1 Overview

Write a C++ class to implement the `BigInt` data type defining signed integers (and operations on these integers) as large as 100 digits.

Here is the `BigInt.h` interface file for your class:

```
#ifndef _BIG_INT_H
#define _BIG_INT_H
namespace EGRE246 {

  class BigInt{
  public:
    static const int MAX_DIGITS = 100;
    enum class Sign {NEG,ZERO,POS};

    BigInt();            // initializes to 0
    BigInt(long long n); // exits program with error if too large
    BigInt(std::string s);    // exits program with error if too large
    int getLen() const;

    int cmp(BigInt& op2) const; // returns -1, 0, or 1

    BigInt abs() const; // absolute value
    BigInt add(BigInt& op2) const;
    BigInt sub(BigInt& op2) const;
    BigInt mul(BigInt& op2) const; // optional
    BigInt div(BigInt& op2) const; // optional
    BigInt mod(BigInt& op2) const; // returns remainder; optional

    friend std::ostream& operator <<(std::ostream& os, BigInt& n);

  private:
    int digits[MAX_DIGITS];
    Sign sign;
    int len; // num of digits not counting sign
  };

}
#endif

bool extra_credit(); // return true if you do extra credit, false otherwise
```

Your arithmetic functions should check for underflow or overflow where appropriate then print an error message and exit the program. You may assume that strings passed to the constructor `BigInt(string)` do not contain any illegal characters (where spaces, a `+` sign, and any non-digits are considered illegal). You will probably need to compile your program with the `-std=c++11` command-line option.

2 Extra Credit

The routines `mul`, `div`, and `mod` are optional but can be done for up to 15% extra credit. You must get all of these extra routines correct to receive the bonus (i.e. it's all or nothing). Everyone should define the function `extra_credit` to indicate if you have done the extra credit routines or not, e.g. implement it (but not as a member function of `BigInt`!) in the following manner:

```
bool extra_credit() {
  return true; // or 'return false' if you did not do the extra credit
}
```

3 Deliverables

You should only turn in your `BigInt` class implementation file. I will test your code with my own driver test program. Name your file `BigInt`*XXXX*`.cpp` where *XXXX* is the last 4 digits of your student id number. For example, if your student id number is V12345678, your file will be named `BigInt5678.cpp` . Projects this term will be submitted via the web using a link off of the class web page (`http://danresler.net/egre246`). Be sure to keep a receipt of your file submission. Note you need not turn in an executable file or your driver program!

4 The `string` C++ Type

Strings in C++ behave in a similar manner as C-style strings except they are not terminated by `'\0'`. Here is a simple demo program illustrating some basic functionality:

```
#include <iostream>
using namespace std;

int main() {
 string s = "12345";
 cout << "s = " << s << endl;
 for (int i = 0; i < s.length(); i++) {
   cout << "s[" << i << "] = " << s[i] << endl;
 }
 int n = s[4]-'0'; // convert a char to a digit
 cout << "s[4] as an int = " << n << endl;
}
```

Due date: Tuesday, February 13