Homework Assignment 4

Due Date: 23:59, April 28, 2016

1 The Big Number

Generally speaking, an "int" variable can present an integer in the range of -2^{31} to $2^{31}-1$. In some cases, it's not enough for us to use. So, we introduce a class `BigNum` (representing "big number") to encapsulate a number which is so large that we cannot simply save it in a simple type. In this assignment, you are asked to implement the class `BigNum` which supports the following functions/operators:

- Member functions:
  1. `BigNum()`
     The default constructor without input, initializes `BigNum`'s value as 0.
  2. `BigNum(int const)`
     A conversion constructor that converts a integer into a `BigNum` object.
  3. `BigNum(const std::string &)`
     A conversion constructor that converts a string representing a large integer into a `BigNum` object.
  4. `BigNum operator+(const BigNum &) const`
     `BigNum operator-(const BigNum &) const`
     `BigNum operator*(const BigNum &) const`
     Overloaded operators that support addition/substraction/mutiplication operations between two `BigNum` objects.
  5. `BigNum& operator++()`
     Overload prefix increment operator.
  6. `BigNum operator++(int)`
     Overload postfix increment operator.
- Global functions:
  1. `BigNum operator+(const BigNum &, const int)`
     `BigNum operator+(const int, const BigNum &)`
     `BigNum operator-(const BigNum &, const int)`
     `BigNum operator-(const int, const BigNum &)`
     `BigNum operator*(const BigNum &, const int)`
     `BigNum operator*(const int, const BigNum &)`
     Overloaded operators that support addition/substraction/mutiplication operation between a `BigNum` object and an integer.
  2. `friend std::ostream &operator<<(std::ostream &, const BigNum &)`
     `friend std::istream &operator>>(std::istream &, BigNum &)`
     Overloaded operators that input/output the number.

Use the attached header file to complete the assignment.

BigNum.h

```
#ifndef BIGNUM_H
#define BIGNUM_H

#include <string>
```

```
#include <iostream>

class BigNum{
        public:
        BigNum();
        BigNum(int const );
        BigNum(std::string const &);
        BigNum operator+(const BigNum &) const;
        BigNum operator-(const BigNum &) const;
        BigNum operator*(const BigNum &) const;
        BigNum& operator++();
        BigNum operator++(int);
        friend std::ostream &operator<<(std::ostream &,const BigNum &);
        friend std::istream &operator>>(std::istream &, BigNum &);

        private:

//Add data feild here to save the data of BigNum.
//Add additional member or global functions to help you set up the class.
};

BigNum operator+(const BigNum &, const int);
BigNum operator+(const int, const BigNum &);
BigNum operator-(const BigNum &, const int);
BigNum operator-(const int, const BigNum &);
BigNum operator*(const BigNum &, const int);
BigNum operator*(const int, const BigNum &);


#endif
```

You have to declare your own private members, for example an array or vector, to store your data. Also you can add additional global or member functions to help you complete this project.

1.1 Input format

Inputs are integers in the range of -10^{50} to 10^{50}represented by strings. If the input is a positive integer or zero, there will be no operand ahead the string. On the other hand, if the input is negative integer, the first char of string will be a '-' operand.

1.2 Output format

Your output should be only the result, without any additional commands or words.

1.3 Examples

test1.cpp

```
#include <iostream>
#include "BigNum.h"
using namespace std;

int main(){
    BigNum a("987654321");
    BigNum b("12345678");
    BigNum c;

    cout << c <<endl;
```

```
    cout << a + b << endl;
    cout << a - b << endl;
    cout << a * b << endl;

    cout << a + 123 << endl;
    cout << a - 123 << endl;
    cout << a * 123 << endl;
}
```

Sample output

```
0
999999999
975308643
12193262222374638
987654444
987654198
121481481483
```

test2.cpp

```
#include <iostream>
#include "BigNum.h"
using namespace std;

int main(){
    BigNum a, b, c;

    cin >> a;

    b=++a;
    cout << a << endl;
    cout << b << endl;

    c=a++;
    cout << a << endl;
    cout << c << endl;
}
```

Sample input

```
999999999
```

Sample output

```
1000000000
1000000000
1000000001
1000000000
```

test3.cpp

```
#include <iostream>
#include "BigNum.h"
using namespace std;

int main(){
    BigNum a("987654321");
```

```
    BigNum b("-987654321");
    BigNum c, d, e;

    cin >> c;

    cout << c << endl;

    cout << a + b << endl;
    cout << a - b << endl;
    cout << a * b << endl;

    cout << b - a << endl;
    cout << b * b << endl;
    cout << b - b << endl;
    cout << b + b << endl;

    d=++c;
    cout << c << endl;
    cout << d << endl;

    e=c++;
    cout << c << endl;
    cout << e << endl;

}
```

Sample input

```
-1000000001
```

Sample output

```
-1000000001
0
1975308642
-975461057789971041
-1975308642
975461057789971041
0
-1975308642
-1000000000
-1000000000
-999999999
-1000000000
```

1.4 Other Notes

- You have to consider :
  1. Initialized input as negative.
     ex. `BigNum` a("-100") ;
  2. BigNum objects calculate with negative integer.
     ex a + -100 ;
  3. The result of calculation is negative.
     ex. a - b == -100 ;
- Don't need to consider to following condition :
  1. The big number objects won't be negative.
     ex. -a , -a + -b

2 How to compile the code at workstation?

Can use this command to compile the code:

```
$ g++ -std=c++14 test1.cpp BigNum.h BigNum.cpp -o main -Wall -Wextra -pedantic -g3
```

3 Report

The report should be limited in 3 pages, with .pdf format. Following contents should be contained :

- Briefly describe how to implement +,-,* operator overloaded.
- Briefly describe how to overload input and output operator.
- What's difference between prefix and postfix increment overloaded? Why they have differnt return type?
- Briefly describe how to handle negative big number operation.

4 Submission

Please name your `.cpp` file as `BigNum.cpp`, `header` file as `BigNum.h`, upload these two files and your report without folder to E3. If you don't follow the rules, you will get no credit.