Assignment 02- A 2nd step of the huge one

Modify your works in the **Assignment 01** to extend its functionality. Replace original add() and subtract() with corresponded operators. Please implement all following relational and arithmetic operations with huge integers: =, ==, !=, <, >, +, -, *, /, %, <<, all of this op are **Binary** operators (<< is made for outputting)
*Use dynamic array or std::vector to obtain higher scores (+20pts). And, use provided driver program (main_hw2.cpp) to test your work. **NOTE:** No floating point and negative value required. All values should be positive integer (include Zero) – (Due date: 4/15)
(Please refer to example_HugeUInteger_hw2.h)

```
#define MAX 1000 //< max length of huge interger
class HugeUInteger
{
    friend ostream &operator<<(ostream &output, const HugeUInteger &in);
public:
    HugeUInteger(); //< default constructor (the default value = 0)
    HugeUInteger(HugeUInteger &in); //< copy constructor
    static HugeUInteger random(unsigned int size); //< randomly generate an integer,
                                               //< the highest digit should not be ZERO
    static HugeUInteger zero(); //< generate a zero number (size = 1)
    HugeUInteger operator=(const HugeUInteger &right); //< assignment
    bool operator==(const HugeUInteger &right); //< equal to
    bool operator!=(const HugeUInteger &right); //< not equal to
    bool operator<(const HugeUInteger &right);  //< less than
    bool operator>(const HugeUInteger &right);  //< greater than

    HugeUInteger operator+(const HugeUInteger &right);
    //< addition operator; HugeUInteger + HugeUInteger
    HugeUInteger  operator-(const HugeUInteger &right);
    //< subtraction operator; HugeUInteger  - HugeUInteger ;
    //< (Use ZERO to replace negative result)
    HugeUInteger  operator*(const HugeUInteger &right);
    //< multiplication operator; HugeUInteger  * HugeUInteger
    HugeUInteger  operator/(const HugeUInteger &right);
    //< division operator; HugeUInteger  / HugeUInteger
    HugeUInteger  operator%(const HugeUInteger &right);
    //< modulus operator; HugeUInteger  % HugeUInteger

private:
///add something to stores a huge integer (dynamic array or std::vector or …)
///add some helper functions
};
```

Sample outputs:

```
Enter seed: 13
Enter the lengths of three huge integers: 7 8 9
N1: 1962866
N2: 93543033
N3: 720778699
N4: 0
N1 + N3 = 722741565
(negative)N2 - N3 = 0
N1 * N4 = 0
N4 = N2 + N3 * N1 = 1414792095334367

Let N2 = N1, then...
N1 is equal to N2
N1 is smaller than N3
N4 / N2 = 15124505
N5 = N4 % N2 = 25010702

N1: 93543033
N2: 93543033
N3: 720778699
N4: 1414792095334367
N5: 25010702
```

A common workflow of preparation of your homework (UPDATED)

(In the following text, we assume your student ID is s1234567 and # is the NUMBER of homework in one digit, Please replace it with your student ID and follow the steps exactly!)

1. **Clone** your private repository of hw# to your home directory on server or another suitable location from our organization on GitHub (https://github.com/YZU-CSE-CS114-Computer-Programming-II)
2. After you start to deal with your homework, **add** a file s1234567hw#_main.cpp as your **main program** and s1234567hw#_report.md as your **report** (optional). Then, **commit** it! (Don't forget to **push** it onto GitHub, too) *.Any of your code should be prefixed with "s1234567hw#_".
   For example: s1234567hw2_ HugeUInteger.h, s1234567hw2_ HugeUInteger.cpp
3. Complete your program and write your report (if necessary) with Markdown.
   *. Please commit your homework when you have any progress.
4. Make sure you have complete the assignment and prepare a clear demo in your report. Do it ON YOUR OWN and ON TIME. (You have only 2 chances in this semester to request additional time for doing your homework, each chance will give you additional **24Hours**. Please contact the TAs **BEFORE** the deadline and use it wisely)