CISP 400 C++ Programming

Project #3

Please note, you are required to include the following when you use g++ to compile: **`-pedantic`**

2. Log on to the Linux server.
3. In a different directory than the one used for project #2, create four files:
    a) ULong.h
    b) ULong.cpp
    c) ULongMain.cpp
    d) Makefile

4. In **`ULong.h`**:

```
#ifndef _ULong_H_
#define _ULong_H_
#include <iostream>
using namespace std;

const unsigned PRECISION = 10000;

class ULong
{
  char      _number[PRECISION];  // index i holds the 10^i digit
  unsigned  _num_digits;

//I used these private member functions in my implementation
  void _initialize();            // _num_digits == 1 &&
                                 // for 0<=i<PRECSION _number[i] == '0'
  void _mult10(unsigned n = 1);  // multiply by 10 n times
  void _mult( char n );          // multiply by n
  void _div10();                 // divide by 10
  bool _is_mult10()const;        // is this a multiple of 10
  void _set_num_digits();


public:

  ULong ( );
  ULong ( const char* );
  ULong ( unsigned long long );
  ULong ( const ULong& );

  ULong& operator= ( const ULong& );
  ULong& operator+= ( const ULong& );
  ULong& operator-= ( const ULong& );
  ULong& operator*= ( const ULong& );
  ULong& operator/= ( const ULong& );
  ULong& operator%= ( const ULong& );

  friend ostream& operator<< ( ostream&, const ULong&);
  friend istream& operator>> ( istream&, ULong&);
  ULong operator+ (const ULong& ) const;
  ULong operator+ (unsigned long long) const;
  friend ULong operator+ (unsigned long long, const ULong&);

  ULong operator- (const ULong& ) const;
  ULong operator- (unsigned long long) const;
  friend ULong operator- (unsigned long long, const ULong&);

  ULong operator* (const ULong& ) const;
  ULong operator* (unsigned long long) const;
  friend ULong operator* (unsigned long long, const ULong&);

  ULong operator/ (const ULong& ) const;
  ULong operator/ (unsigned long long) const;
  friend ULong operator/ (unsigned long long, const ULong&);

  ULong operator% (const ULong& ) const;
  ULong operator% (unsigned long long) const;
  friend ULong operator% (unsigned long long, const ULong&);

  ULong operator++ (int); //post
  ULong operator-- (int); //post
  ULong& operator++ (); //pre
  ULong& operator-- (); //pre

  bool operator== (const ULong& ) const;
  bool operator== (unsigned long long) const;
  friend bool operator== (unsigned long long, const ULong&);

  bool operator< (const ULong& ) const;
  bool operator< (unsigned long long) const;
  friend bool operator< (unsigned long long, const ULong&);

  bool operator!= (const ULong& ) const;
  bool operator!= (unsigned long long) const;
  friend bool operator!= (unsigned long long, const ULong&);

  bool operator> (const ULong& ) const;
  bool operator> (unsigned long long) const;
  friend bool operator> (unsigned long long, const ULong&);

  bool operator<= (const ULong& ) const;
  bool operator<= (unsigned long long) const;
  friend bool operator<= (unsigned long, const ULong&);

  bool operator>= (const ULong& ) const;
  bool operator>= (unsigned long long) const;
  friend bool operator>= (unsigned long long, const ULong&);

};

#endif
```

In **`ULong.cpp`**, implement these member functions. We will standardize the implementation for this assignment. The value in our internal array of char at index i will represent the 10^ith digit in the number. For example, if we were to store the number 987, `_number[0] == ‘7’, _number[1] == ‘8’, and _number[2] == ‘9’` Notice that we are storing the numbers as characters

YOU MUST NOT USE ANY BUILT-IN FUNCTIONS

YOU MUST IMPLEMENT THE FUNCTIONS IN YOUR .CPP FILE IN THE SAME ORDER AS I HAVE LISTED THEM IN THE .H FILE

For the comparison operator you must implement two and only two of them directly:

```
bool operator== (const ULong& ) const;
bool operator< (const ULong& ) const;
```

All other comparison operators **MUST** be implemented in terms of these two. For example:

```
bool operator!= (const ULong& L ) const
{
  return !(*this == L);
}
```

All binary arithmetic operators must be implemented in terms of the corresponding immediate operators, i.e.,

```
ULong operator+ (const ULong& ) const;
```

Must be implemented in terms of

```
ULong& operator+= (const ULong& );
```

In **`ULongMain.cpp`**, write a test main to test your code.

Submit these four files via the drop-box