```cpp
#define _CRTDBG_MAP_ALLOC
#include <stdlib.h>
#include <crtdbg.h>

#ifdef _DEBUG
#ifndef DBG_NEW
#define DBG_NEW new ( _NORMAL_BLOCK , __FILE__ , __LINE__ )
#define new DBG_NEW
#endif
#endif // _DEBUG
//-----------------------------------------------------------
#include <iostream>
using namespace std;
#include "BigInt.h"
#include <string.h>
//-----------------------------------------------------------

//The n'th Fibonacci number n(1)=1,n(2)=1,n(3)=2,n(4)=3,n(5)=5,...
BigInt fibo(unsigned int n)
{
    //in case i dont need to calculate the organ of fibonacci series:
    BigInt n_fib_number = BigInt_create(0);

    if (n == 1 || n == 2)
    {
        BigInt_inc(n_fib_number);
    }
    else if (n > 2)  // in case i need to calculate the organ of fibonacci series , must initialize fibonacci series and then calculate:
    {
        //initialize fibonacci series:
        BigInt a = BigInt_create(1);
        BigInt b = BigInt_create(1);
        BigInt tmp;
        //calculating The n'th Fibonacci number
        for (unsigned int i = 2; i < n; i++)
        {
            tmp = a + b;
            BigInt_assign(n_fib_number, tmp);
            BigInt_assign(a, b);
            BigInt_assign(b, n_fib_number);
            BigInt_destroy(tmp);
        }
        BigInt_destroy(a);
        BigInt_destroy(b);
    }

    return n_fib_number;
}

//sqrt(x)
BigInt sqrt(const BigInt& x)
{
```

```
    BigInt sum = BigInt_create(0);
    BigInt odd = BigInt_create(1);
    BigInt sq = BigInt_create(0);
    BigInt const2 = BigInt_create(2);
    BigInt tmp;
    while (sum < x)
    {
        tmp = sum + odd;
        BigInt_assign(sum, tmp);
        BigInt_destroy(tmp);

        BigInt_inc(sq);

        tmp = odd + const2;
        BigInt_assign(odd, tmp);
        BigInt_destroy(tmp);
    }
    BigInt_destroy(sum);
    BigInt_destroy(odd);
    BigInt_destroy(const2);

    return sq;
}

////////////////////////////////////////////////////////////////
//operator overload:

int operator<(const BigInt& a, const BigInt& b)
{
    int ans = BigInt_compare(a, b);
    if (ans == -1)
        ans = 1;
    else
        ans = 0;

    return ans;
}

int operator==(const BigInt& a, const BigInt& b)
{
    if (BigInt_compare(a, b) == 0)
        return 1;
    else
        return 0;
}

BigInt operator+(const BigInt& a, const BigInt& b)
{
    return BigInt_add(a, b);
}
```