In this lab, we will discuss how to code your own tester files that will enable you to correctly and precisely test your code.

1. Testing the Constructors
    a. Default constructor

        The default constructor is the constructor that the compile will execute when there's no parameter is given. In your assignment, the default constructor should initialize the BigNum as zero because nothing is specified.

        BigNum testDefaultConstructor = new BigNum();

        How do you test if this is actually happening? Check that in the BigNum.h file, the function printBigNum converts digits array into a string of characters. So use this string returned by the printBigNum function and compares that to the string "0" and check if it works. If it doesn't, then fix the code where it needs to be fixed.

        Use strcmp function to check if two strings are equal. It is declared in the cstdlib.

        strcmp(testDefaultConstructor.printBigNum(), "0")

        This function returns 0 if two strings are equal.

        strcmp ("abc","abc") returns 0
        strcmp("ed","wefw") returns a non zero value

    b. Integer Constructor:

        This constructor takes in an integer and then stores that in the digits array. Testing of this constructor is the same as above apart from the fact that now when declaring and initializing the Bignum Variable, you call this particular constructor.

        BigNum testIntConstructor = new BigNum(100);

        Repeat the code you did for the default constructor and check if your constructor is foolproof.

        Critical inputs:
            a. check for a negative integer
            b. check for zero

c. **check for any random integer**
d. **check for an integer that starts with a '+' (+123)**

c. String Constructor:

This Constructor takes in a string of numerical digits and stores it up in the Bignum's digits array. Test the constructor by repeating the above mentioned process.

BigNum testStringConstructor = new BigNum("12334");

Critical Inputs:

a. **Check for a big positive number**
b. **Check for a big negative number**
c. **Check for zero**
d. **Check for negative zero**
e. **Check for a positive number with leading zeros**
f. **Check for a negative number with leading zeros**

d. Copy Constructor:

This Constructor takes in another BigNum instance and copies up the digits array of that constructor into its own.

Examples for constructor check:

// this is the sample code for checking the string constructor whether it works //when it is initialized with "0"

```
BigNum test1 = new BigNum ("0"); // initialize the BigNum with 0
 if(strcmp(test1.printBigNum(),"0")==0) // check if the test1 BigNum
//contains 0
        cout<< "The test has passed";
 else
        cout<<"The test has failed";
```

Now try to add more and more tests in your partner's tester file with different critical inputs that you think might be able to break your partner's constructor code. If you can then kudos! And if you can't well kudos to your partner then!

2. Examples for assignment operation check:
   Here you have to test if assignment operations of two(a=b) or more instances(a=b=c) of BigNum class is working or not.

   Check all possible combinations of Bignum instances created by all the constructors (and also check with different corresponding critical inputs) to break the code of your partner.

// this is a sample code that will check whether the assignment operation is working //accordingly.

```
BigNum test1 = new BigNum (“0”);//String Constructor
BigNum test2 = new BigNum (1234);//int constructor
 test1 = test2; // assign test 2 to test1
if(strcmp(test1.printBigNum (),test2.printBigNum())==0)
        cout<<”test has passed”; // test1 should contain test2’s bignum
 else
        cout<<”test has failed”;
```

Now try to check assignments like this (test1=test2=test3) and then check if that works too. In this case, both test2 and test1 should contain test3’s bignum.

3. Examples for checking operator codes. (+=,-=,*=,/=,%=)

   For these overloading functions, declare two Bignum instances initialized with critical inputs and then check the output.
   For example

   Bignum test1;
   BigNum test2;
    test1+=test2;

   ///now test1 should contain 0 because both test1 and test2 were created using the default constructor. Same goes for the other operators.

   // this is a sample code for testing += operator test.

```
BigNum test1 = new BigNum (“123”);
BigNum test2 = new BigNum (456);
  test1+=test2;
if(strcmp(test1.printBigNum(),”579”)==0)
        cout<< “test has passed”; // test1 should contain the result

else
        cout<<”test has failed”;
```

   Now try to add more tests like this with critical inputs to break the code of your partner.

4. Examples for checking more operator operations.
   Here check if the operation is working accordingly. Here the resultant of the operation will be stored in a third instance of BigNum(see the code).

//sample for checking +,-,*,/,% operators

```
BigNum test1 = new BigNum ();
BigNum test2 = new BigNum("0");
BigNum test3 = test1+test2;
 if(strcmp(test3.printBigNum(),"0")==0)
        cout<<"test has passed";
 else
        cout<<"test has failed";
```

5. Examples for checking logical operators ==,!=,<=,>=,<.>
   Check the codes for the logical operators.

   //sample code for checking <= operator

```
BigNum test1 = new BigNum ("0");
BigNum test2 = newBigNum(0);
 if(test1 <= test2)
        cout<<"test has passed";
else
        cout<<"test has failed";
```

   Check for different critical inputs and see whether your partner's code reacts accordingly or not.

6. ++,--

   Here the Bignum instance's number should be incremented or decremented by 1.

   //sample code

```
BigNum test1 = new BigNum ("0");
 test1++;
if(strcmp(test1.printBigNum (),"1")==0)
        cout<<"test has passed";
else
       cout<<"test has failed";
```