Scientific calculation often needs to operate values with many digits or with high precision. However, the built-in data types only provide a fixed size for representing values. A common way to address the issue is to provide a class for big integer, which can manipulate integers with arbitrary number of digits.
Your goal is to provide a class BigInt for representing the big integer with a string of digits and a sign. The class BigInt should provide the following functionalities:

1. Overload stream insertion operator (<<) to output the class object.
2. Overload stream extraction operator (>>) to input the class object.
3. Provide proper constructors to initialize an object from a string, and from an long long integer.
4. Provide copy constructor.
5. Overload addition operator (+) to add two class objects.
6. Overload subtraction operator (-) to subtract two class objects.
7. Overload relational operators (> and <), and equality operators (== and !=) to compare two class objects.

The above overloaded operators should behave as if they perform on the built-in integer types.
The implementation of addition and subtraction can be simplified as two cases A + B and A – B, where A and B are both positive integers. For addition, we have:
-A + B = B – A, A + (-B) = A – B, -A + (-B) = -(A+B),
and for subtraction, we have:
-A - B = -(A+B), A – (-B) = A + B, -A – (-B) = B – A
Therefore the eight cases of addition and subtraction are reduced to two cases, and the sign of the result is determined by the magnitude.
Requirement: Use the sample main function to complete your program. Provide a class Banker satisfying all of the above conditions and separate the interface (BigInt.h) and the implementation (BigInt.cpp).
Prohibited: Use C-style input/output.

Input

Each case contains two integers n_1 and n_2. The input ends with an asterisk (*).

Output

For each case, output the two input big integers, check and output the relation of the two big integers, and output their sum (n_1 + n_2) and difference (n_1 - n_2).

Sample Input

```
-80948824551711527614232216857618927954312334113874277931986502860248 65
-80613893446066184963788291359840763615420973726016575412001460 71

46182119762952003918195325258677229419698255491250839396 7
-56935766582544161633553282536186214629150364929344059634288 7581

*
```

Sample Output

```
Big Integer 1: -80948824551711527614232216857618927954312334113874277931986502860248 65
Big Integer 2: -80613893446066184963788291359840763615420973726016575412001460 71
Big Integer 1 is less than Big Integer 2.
The sum of them is: -80948905165604973680417180645910287795075949534848003948561914861709 36
The difference of them is: -80948743937818081548047253069327568113548718692900551915411090858787 94

Big Integer 1: 46182119762952003918195325258677229419698255491250839396 7
Big Integer 2: -56935766582544161633553282536186214629150364929344059634288 7581
Big Integer 1 is greater than Big Integer 2.
The sum of them is: -56935720400424398681549364340860955951920945231088568383449361 4
The difference of them is: 56935812764663924585557200731511473306379784627599550885128154 8
```