1. “Briefly describe how to implement +,-,* operator overloaded.”

In the syntax point of view, for member functions, we declare the operations inside the class definition and define them outside the class with “BigNum::” to specify its scope. If we define them as global function, we use “friend” to tag them as friend function; then, declare and define under the global scope. By the way, we use keyword “operator” to overload an operator.

In the arithmetic point of view, firstly for operation “+”, we add each digit of both operand iteratively. Then, right after the addition of each digit, we perform carrying operation. Above is how we implement addition in which the operands are restricted to be positive. As for varying conditions of different positivity of operands, we implement a subtraction function “sub” for big-minus-small in premise that both operands are positive. When we encounter addition with both operands negative, we set the outcome negative and add their magnitude together. When we encounter addition with one negative and one positive operands, we first identify the one with greater magnitude. The positivity of the greater-magnitude operand is the positivity of the outcome value. And the magnitude of them is the value we get after we subtract the magnitude of both operands.

For multiplication, we do exclusive-nor operation to decide its positivity first. Then multiply each digit; finally, we perform carrying operation.

For subtraction, we do it by calling addition operation after changing the positivity of its right operand.

2. “Briefly describe how to overload input and output operator.”

We declare and define them as global friend functions of BigNum in the syntax point of view.

In the arithmetic point of view, for output operator, we cut out the leftmost zeros, printing out ‘-‘ according to its positivity, and then print out from the first nonzero from left. For input operator, we simply input to a string then use constructor and assignment to send in the “BigNum” value.

3. “What’s difference between prefix and postfix increment overloaded? Why they have different return type?”

In the syntax point of view, prefix increment need no int parameter, postfix increment, however need one. Moreover, a prefix increment usually returns a reference of the object, but a postfix increment don’t.

4. “Briefly describe how to handle negative big number operation.”

Basically we transform each “+” or “-” operation into one basic “+” operation, the operation that can handle addition among both negative and positive “BigNum.” For subtractions, we just change the positivity of the right operands; then, perform the addition. In that addition, when we encounter negative, there’re two situations. Both

are negative or merely one of them are negative. For the former situation, we add the magnitude of both operand together and have the output positivity be negative. For latter situation, we subtract the magnitude of the operand with smaller magnitude from the magnitude of the operand with bigger magnitude as the result magnitude, and we set the outcome positivity be the positivity of the operand with greater magnitude.