Project Description: Binary Number Calculator

Abstract data type: Binary

The main part of this assignment is to create a class called Binary implementing an abstract data type for binary numbers of arbitrary length (no limit to the number of bits). Implement the internal representation as an array of bool values, where true represents a 1 and false represents a 0. Each binary number is stored in one array that is just large enough to hold the most significant bit in the number. Initially, a constructor will dynamically allocate the array to store just the number of bits necessary for the number, with a minimum of one bit. Your code should handle both positive and negative numbers (use a separate bool in the private data to simulate the sign bit). Do not use any floats or doubles in your code, including floating point operations, such as log() or pow(). Do not convert the number to an integer except where noted below. Do not store the integer in your private data.

I want you to develop a full class, including all of the following methods:

* Constructors and destructors
* Default constructor (set the number to zero)
* One parameter constructor that takes in the number as an integer and converts it to binary
* Copy constructor
* Destructor
* Overloaded operators
* Addition and subtraction (operator+ and operator-): For these two operations only, you can convert the internal binary to an integer.
* Assignment (operator=)
* Addition and subtraction assignment (operator+= and operator-=)
* Equality and inequality (operator== and operator!=)
* Accessors
* Output (operator<<): Output the binary number with no spaces. Use a preceding "-" if it is negative. Do not output leading zeros. Do not output endl.
* Get a single bit (getBit): returns a bool corresponding to the correct bit. Bit 0 is the least significant bit. (For example: bool bit = n.getBit(3);
* Mutators
* Input (operator>>) that takes an integer from the keyboard
* setBit(n): sets bit n to one/true
* clearBit(n): sets bit n to zero/false
* toggleBit(n): changes false to true and true to false.

As always expected when programming, comment clearly and thoroughly. Comments in the class files should describe the ADT, all functionality, and assumptions. Functions in the .cpp file must be separated by white space. Each function should have documentation, including pre-conditions and post-conditions listed at both declaration and implementation. Internal comments should be used whenever it is not obvious what the code is doing. For more information, see the Program grading rubric.

You must handle errors such as trying to access an invalid value, but you do not need to worry about the input data type for operator>> (you will not get non-integer input, e.g., a string.) All operations must work correctly and not crash on any other valid input. Do not print error messages, do not throw exceptions, and do not exit the program – deal with errors in some other appropriate manner (usually ignoring them).

You will turn in your code in a zip file containing (only) Binary.h and Binary.cpp.

Your program will be tested with Microsoft Visual C++ Community Edition (2019). You should test your code with this compiler before turning it in. Here is a simple driver that your code should compile with: HW1.cppPreview the document. I will be testing with a much more demanding version, so test your code thoroughly.