The codes for the “parameter constructor” and “binary operator +” are provided below. Note that these codes are based the student’s work from the last semester. These codes are NOT meant to be the “best codes”, but hopefully they can provide you (especially those students who need help) with some hints and/or ideas. Please do not blindly follow these codes. Instead, try to get ideas and work on creating your codes. It is your responsibility to make these codes working, if you decide to re-use them, since there is possibility that the internal algorithms might not be the exactly same as your own algorithms. The reasons that these two codes are provided: (1) the parameter constructor is the base (i.e., the starting point), meaning that without this part working correctly, it will be difficult to work on other parts, (2) providing the binary operator “+” is that there are two other operators (“-“ and “\*”) which are likely more complex than the “+” operator. But, you can get some idea for the implementation of them.

Parameter Constructor : Binary (int x)

Binary Operator “+”

Binary::Binary(int x)

{

firstTerm = nullptr;

unsigned degree;

while (x > 0)

{

// The degree of the next bit is the integer value of the base-2 log of x, where the base-2 log of x is log(x) / log(2).

degree = unsigned(log(float(x)) / log(float(2)));

// Create a BinaryNode with degree.

set\_bit(1, degree);

x -= pow((float)2, (float)degree);

}

}

Binary operator+(const Binary &b1, const Binary &b2)

{

Binary newBinary = Binary();

unsigned iMax = max(b1.get\_degree(), b2.get\_degree());

// The carry bit

unsigned bCarry = 0;

// Traverse each bit of b1 or b2 (whichever has the higher degree).

for (unsigned i = 0; i <= iMax; ++i)

{

unsigned sum = b1.get\_bit(i) + b2.get\_bit(i) + bCarry;

// Determine if there is a new carry bit.

if (sum >= 2)

{

bCarry = 1;

sum -= 2;

}

else

{

bCarry = 0;

}

if (1 == sum)

{

newBinary.set\_bit(1, i);

}

}

// Handle the extra carry bit.

if (1 == bCarry)

{

newBinary.set\_bit(1, iMax + 1);

}

return newBinary;

}