DIVIDE TWO INTEGERS

☆ In this problem, we are given two non zero integers and our job is to divide them without using any in built operator.

We are also supposed to keep int value overflow in mind, as -2^{31} and -1 divided would result in 2^{31} which cannot be stored in a 32 bit integer.

Bruteforce solution is to obviously loop subtractions from dividend to divisor. Time Complexity is O(dividend) which is too much.

Since subtractions one by one are very slow, so we will do subtractions by multiplying the divisor with maximum possible power of 2.

Pseudocode :

```
divide (dividend, divisor) {
    if (dividend == divisor) {
        return 1
    }
    signPos = true
    if (dividend >= 0 & divisor < 0) {
```

```
        sign = false
    } else if (dividend < 0 && divisor > 0) {
        sign = false
    }
    ans = 0
    n = abs(dividend)
    d = abs(divisor)
    while (n >= d) {
        c = 0
        while (n >= d * 2^(c+1)) {
            c++
        }
        ans += 2^c
        n = n - (d * 2^c)
    }
    if (ans >= INT_MAX && signPos) {
        return INT_MAX
    } else if (ans >= INT_MAX &&
                                    !signPos){
        return INT_MIN
    } else {
        return ans
    }
}
```