## → Large exponentiation using Binary Multiplication

-> Let's suppose in the last problem we have: ~~$a \leq 10^{18} \leq M \leq 10^{18}$~~

~~$a \leq 18$~~ $a \leq 10^{18} \leq M \leq 10^{9}$

∴ We can take $a = a \,\%\, M$ in the beggining of function to prevent it from overflow. But what if, we have $a \leq 10^{18} \leq M \leq 10^{18}$

Now, there will be a case if we will be doing:

```
LL ANS = (A * A) %. M
          ↓   ↓
        10^18 10^18
```

This will still overflow because we can't

directly multiply $10^{18} \times 10^{18}$.
So, to solve such kind of problems we use Binary Multiplication.

∴ What is exactly a * a ?
→ It is a added a times.
a + a + a ——— a times.

So, to prevent overflow instead of directly multiplying, we can take %M after each adding.

e.g ~~(a * a) %M~~

$a + a < 2 \times 10^{18}$ (∵ $a = 10^{18}$)
$\%M < 10^{18}$

$+a < 2 \times 10^{18}$
$\%M < 10^{18}$ ∴ It will never overflow

∴ To achieve this we can use Brute Force Method, which will be the using of FOR loop till $a \leq 10^{18}$.
But it's T.C is O(a).

→ We can achieve this in O(log(n)) T.C using Binary Multiplication, (similar to Binary exponentiation of iterative method).

Suppose, we have a = 3 and b = 13.,
We have to do a x b.
We can achieve this using binary multiplication like this:

| | |
|---|---|
| 3 (13) | 13 → 1 1 0 1 (In binary) |
| 3 (8+4+0+1) | => (8+4+0+1) |
| 3 → 3 | (3 x 1) We want this |
| 6 | (3 x 2) We don't want this |
| 12 → 15 | (3 x 4) We want this |
| 24 → (39) | (3 x 8) We want this. |

```
INT BINMULTIPLY (LL A, LL B)
{
    LL ANS = 0;
    WHILE (B > 0)
    {
        IF (B & 1)
        {
            ANS = (ANS + A) %. M;
        }
        A = (A + A) %. M;
        B >>= 1;
    }
    RETURN ANS;
}
```