Yuan Gao z5239220 Q1

Given positive integers M and n compute M^n using only $O(\log n)$ many multiplications.

We can take the divide and conquer method to solve it.
For instance,

M^{75} can be divided to $M^{75} = M \cdot M^2 \cdot M^8 \cdot M^{64} = M^{2^0} \cdot M^{2^1} \cdot M^{2^3} \cdot M^{2^6}$

At the same time，
The binary of 75 is 1001011

2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	0	0	1	0	1	1

Therefore,
We can calculate M^n by binary number

$M^n = M^{2^{i_0}} \cdot M^{2^{i_1}} \cdot M^{2^{i_3}} \dots\dots M^{2^{i_k}}$

There is C++ code:

```cpp
long long myPow(int M, int n){
    /*set result */
    long long res = 1;
    while(n > 0){
        /*Bit operation, if the last bit is 1
        do the res *= M*/
        if((n & 1)){
            res *= M;
        }
        M *= M;
        /*right shifted one time*/
        n = n >> 1;
    }
    return res;
}
```

Because M stored the last product in a while loop, the time complexity is $O(\log n)$