z5173405 JIE MEI

Question 1

Firstly, we need to convert n to binary code. $n = 2^{k_1} + 2^{k_2} + \cdots 2^{k_m}$, $k_1 = \lfloor \log_2 n \rfloor$ where $k1 > k2 > \cdots > km$; then we will get $M^n = M^{2^{k_1}} * M^{2^{k_2}} \ldots M^{2^{k_m}}$ that involves at most $\lfloor \log_2 n \rfloor$ multiplications. when we want to calculate M^n ,we can first calculate $y = M^{\lfloor \frac{n}{2} \rfloor}$. According to the result of recursive calculation, if n is even, $M^n = y^2$. If n is odd, $M^n = y^2 * M$. Since each recursion will halve the exponent, so at most $\lfloor \log_2 n \rfloor$ multiplications.

Sample code(Java)

```java
class Solution {
    public static int myPow(int m, int n) {
        long b = n;
        int res = 1;
        if(b < 0) {
            m = 1 / m;
            b = -b;
        }
        while(b > 0) {
            if((b & 1) == 1) res *= m;
            else {
                m *= m;
                b >>= 1;
            }
        }
        return res;
    }

    public static void main(String[] args) {
        int res = myPow(2,16);
        System.out.println(res);
    }
}
```