# Q1. <https://leetcode.com/problems/powx-n/>

## Solution:

*/\*\*  
 \* @author pranoy.chakraborty  
 \* @Date 11/06/2023  
 \*/*public class PowerOfN {  
 public static void main(String[] args) {  
 System.*out*.println(*myPow*(2.0000, 10));  
 System.*out*.println(*myPow*(2.000,-2));  
 }  
  
 static double myPow(double x, int n) {  
 double temp;  
 if (n == 0) return 1;  
 temp = *myPow*(x, n / 2);  
 if (n % 2 == 0) {  
 return temp \* temp;  
 } else if (n > 0) {  
 return x \* temp \* temp;  
 } else {  
 return temp \* temp / x;  
 }  
 */\*if (n == 0) {  
 return 1;  
 }  
 return x \* myPow(x, n - 1);\*/* }  
  
}

# Q2. <https://leetcode.com/problems/power-of-two/>

## Solution*:*

*/\*\*  
 \* @author pranoy.chakraborty  
 \* @Date 11/06/2023  
 \*/*public class PowerOfTwo {  
 public static void main(String[] args) {  
 System.*out*.println(*isPowerOfTwo*(16));  
 System.*out*.println(*isPowerOfTwo*(3));  
 }  
  
 static boolean isPowerOfTwo(int n) {  
 *//method 1: using bitwise operation  
 //return (n > 0) && ((n & n - 1) == 0);  
  
 //method 2: using recursion* if (n == 1) {  
 return true;  
 } else if (n % 2 != 0 || n == 0) {  
 return false;  
 } else {  
 return *isPowerOfTwo*(n / 2);  
 }  
 }  
}

# Q3. <https://leetcode.com/problems/count-good-numbers/>

## Solution:

*/\*\*  
 \* @author pranoy.chakraborty  
 \* @Date 13/06/2023  
 \*/*public class CountGoodNumbers {  
  
 static long *mod* = (int) (1e9 + 7);  
  
 public static void main(String[] args) {  
 System.*out*.println(*countGoodNumbers*(4));  
 System.*out*.println(*countGoodNumbers*(50));  
 }  
  
 static int countGoodNumbers(long n) {  
 long even = (n + 1) / 2;  
 long odd = n / 2;  
 return (int) ((*myPow*(5, even)) \* (*myPow*(4, odd)) % *mod*);  
 }  
  
 static long myPow(long x, long y) {  
 long temp = 1;  
 if (y == 0) return 1;  
 temp = *myPow*(x, y / 2);  
 if (y % 2 == 0) {  
 return (temp \* temp) % *mod*;  
 } else {  
 return (x \* temp \* temp) % *mod*;  
 }  
 }  
}

# Q4. <https://leetcode.com/problems/minimum-non-zero-product-of-the-array-elements/>

## Solution:

*/\*\*  
 \* @author pranoy.chakraborty  
 \* @Date 13/06/2023  
 \*/*public class MinimumProductOfNonZeroElement {  
  
 public static int *mod* = (int) (1e9 + 7);  
  
 public static void main(String[] args) {  
 System.*out*.println(*minNonZeroProduct*(2));  
 System.*out*.println(*minNonZeroProduct*(3));  
 System.*out*.println(*minNonZeroProduct*(54));  
 }  
  
 static int minNonZeroProduct(int p) {  
 long max = (long) (Math.*pow*(2, p) - 1);  
 long power = *myPow*(max - 1, (max - 1) / 2);  
 power = ((power % *mod*) \* (max % *mod*)) % *mod*;  
 return (int) power;  
 }  
  
 static long myPow(long x, long y) {  
 long temp = 1;  
 if (y == 0) return 1;  
 temp = *myPow*(x, y / 2);  
 temp = temp % *mod*;  
 if (y % 2 == 0) {  
 return (temp \* temp) % *mod*;  
 } else {  
 return (((temp \* temp) % *mod*) \* (x % *mod*)) % *mod*;  
 }  
 }  
}