# 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/valid-anagram/

## Solution:

import java.util.Arrays;  
  
*/\*\*  
 \* @author pranoy.chakraborty  
 \* @Date 18/05/2023  
 \*/*public class QuestionThreeSolution {  
 public static void main(String[] args) {  
 System.*out*.println(*isAnagramMethod1*("anagram", "nagaram"));  
 System.*out*.println(*isAnagramMethod2*("anagram", "nagaram"));  
 }  
  
 *//time complexity -> O(nlogn)* static boolean isAnagramMethod1(String s, String t) {  
 char[] sArray = s.toCharArray();  
 Arrays.*sort*(sArray);  
 char[] tArray = t.toCharArray();  
 Arrays.*sort*(tArray);  
 return Arrays.*equals*(sArray, tArray);  
 }  
  
 *//time complexity -> O(n), space complexity -> O(1)* static boolean isAnagramMethod2(String s, String t) {  
 int[] freq = new int[26];  
 for(char c: s.toCharArray()){  
 freq[c-'a']++;  
 }  
 for(char c:t.toCharArray()){  
 freq[c-'a']--;  
 }  
 return Arrays.*stream*(freq).allMatch(x->x==0);  
 }  
}

# Q4. https://leetcode.com/problems/ugly-number/

## Solution:

*/\*\*  
 \* @author pranoy.chakraborty  
 \* @Date 18/05/2023  
 \*/*public class QuestionFourSolution {  
 public static void main(String[] args) {  
 System.*out*.println(*isUgly*(6));  
 System.*out*.println(*isUgly*(14));  
 System.*out*.println(*isUgly*(8));  
 }  
  
 *//time complexity -> O(logn)* static boolean isUgly(int n) {  
 if (n == 0) return false;  
 for (int i = 2; i <= 5; i += i - 1) {  
 while (n % i == 0) {  
 n /= i;  
 }  
 }  
 return n == 1;  
 }  
}