

Description

 Editorial Solutions Submissions

172. Factorial Trailing Zeroes

Solved

Medium

 Topics Companies

Given an integer n , return the number of trailing zeroes in $n!$.

Note that $n! = n * (n - 1) * (n - 2) * \dots * 3 * 2 * 1$.

Example 1:

Input: $n = 3$
Output: 0
Explanation: $3! = 6$, no trailing zero.

Example 2:

Input: $n = 5$
Output: 1
Explanation: $5! = 120$, one trailing zero.

Example 3:

Input: $n = 0$
Output: 0

Constraints:

- $0 \leq n \leq 10^4$

Code

Java

 Auto

```
1 class Solution {
2     public int trailingZeroes(int n) {
3         int count = 0;
4         if (n<5)
5             return 0;
6         while (n>=5){
7             count = count + n/5;
8             n= n /5;
9         }
10        return count;
11    }
12 }
```

Saved

Ln 1, Col 1

Testcase

 Test Result

Accepted

 Runtime: 0 ms

Case 1 Case 2 Case 3

Input

n =
3

Output

0

Expected

0

[Description](#) |
 [Editorial](#) |
 [Solutions](#) |
 [Submissions](#)

231. Power of Two

Solved

[Easy](#)
[Topics](#)
[Companies](#)

Given an integer `n`, return `true` if it is a power of two. Otherwise, return `false`.

An integer `n` is a power of two, if there exists an integer `x` such that `n == 2x`.

Example 1:

Input: `n = 1`
Output: `true`
Explanation: $2^0 = 1$

Example 2:

Input: `n = 16`
Output: `true`
Explanation: $2^4 = 16$

Example 3:

Input: `n = 3`
Output: `false`

Constraints:

Code

Java Auto

```

1 class Solution {
2     public boolean isPowerOfTwo(int n) {
3         if (n <= 0) {
4             return false;
5         }
6         return (n & (n-1)) == 0;
7     }
8 }
    
```

Saved

Ln 1, Col 1

☒ Testcase |
 Test Result

Accepted Runtime: 0 ms

☒ Case 1
 ☒ Case 2
 ☒ Case 3

Input

`n =`
`1`


Output

`true`

Expected

`true`

258. Add Digits

Solved 
[Easy](#) | [Topics](#) | [Companies](#) | [Hint](#)

Given an integer `num`, repeatedly add all its digits until the result has only one digit, and return it.

Example 1:

Input: `num = 38`

Output: `2`

Explanation: The process is

`38 → 3 + 8 → 11`

`11 → 1 + 1 → 2`

Since 2 has only one digit, return it.

Example 2:

Input: `num = 0`

Output: `0`

Constraints:

- $0 \leq \text{num} \leq 2^{31} - 1$

Follow up: Could you do it without any loop/recursion in $O(1)$ runtime?

Code

Java  Auto

```

1 class Solution {
2     public int addDigits(int num) {
3         if(num == 0){
4             return 0;
5         }
6         return 1+((num-1)%9);
7     }
8 }

```

Saved

Ln 1, Col 1

☒ Testcase | [Test Result](#)

You must run your code first