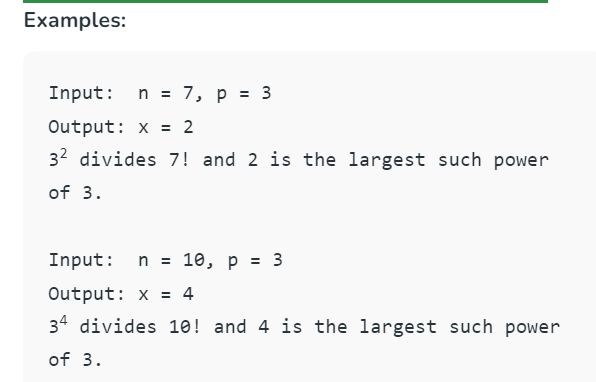
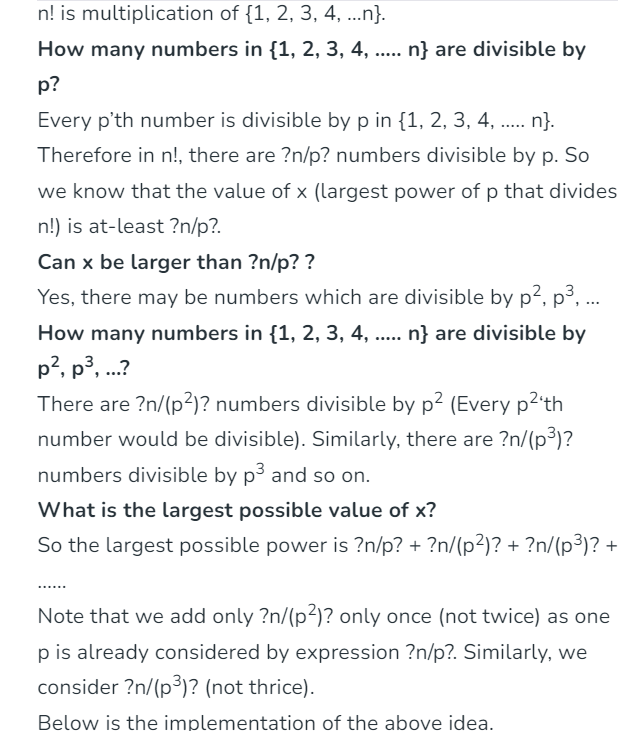
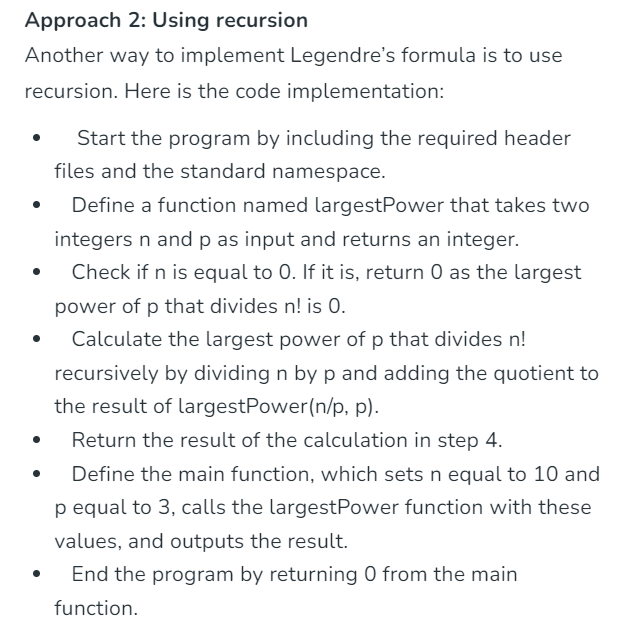
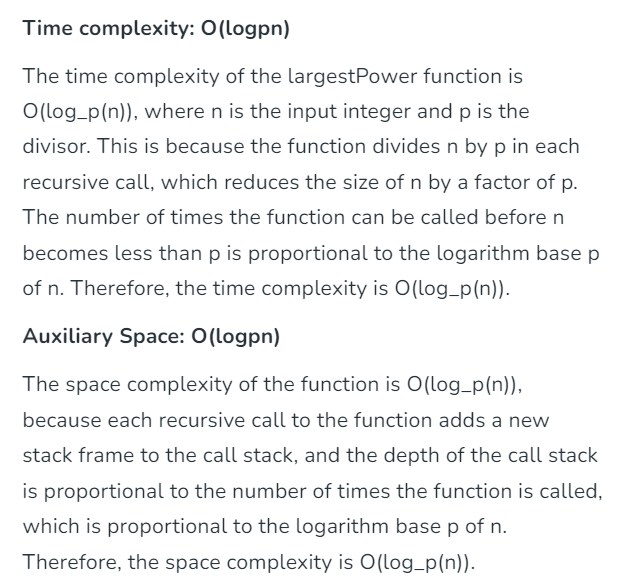
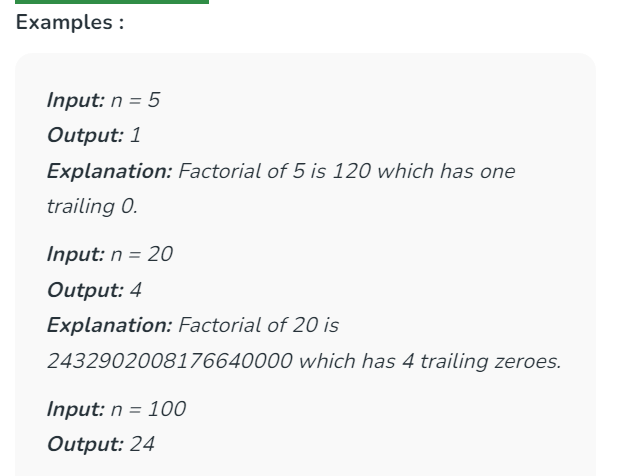
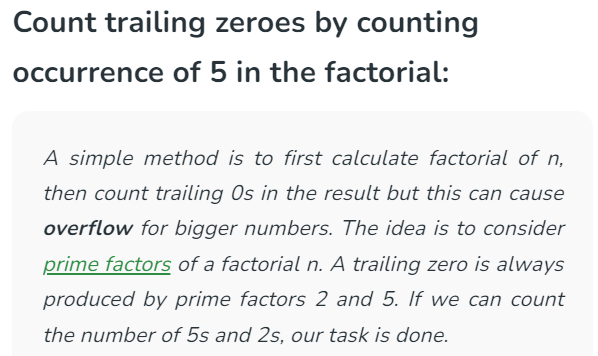
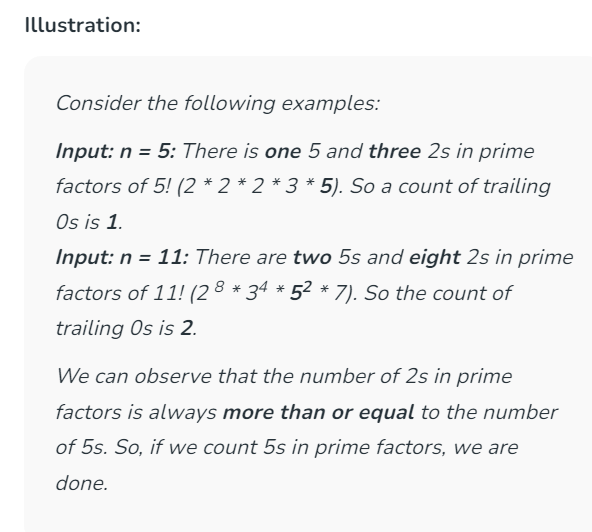
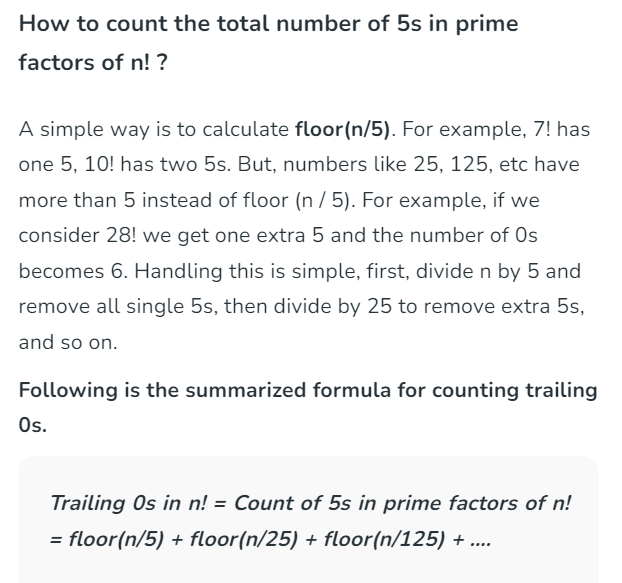
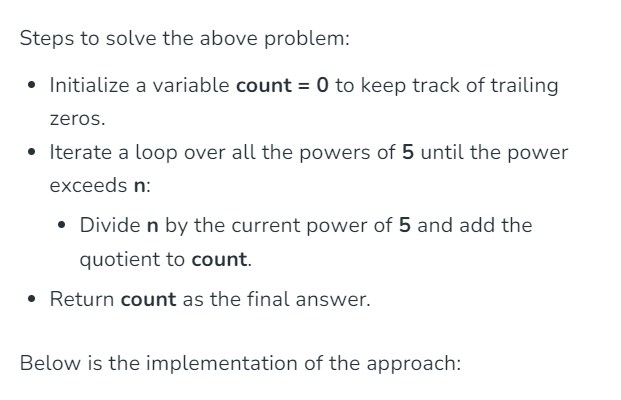
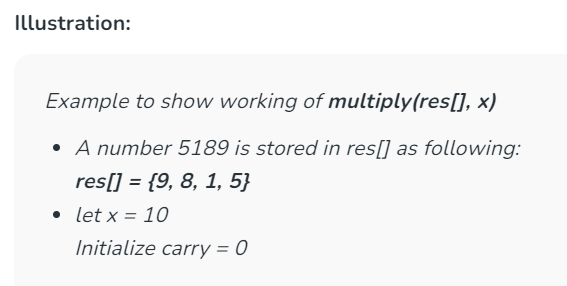
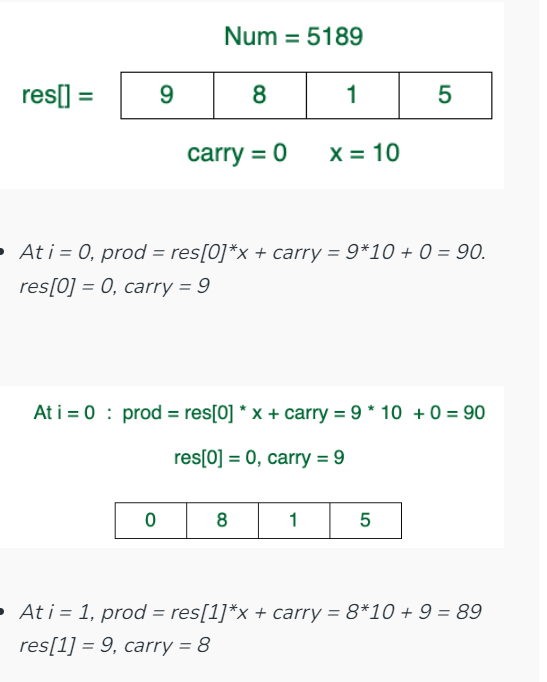
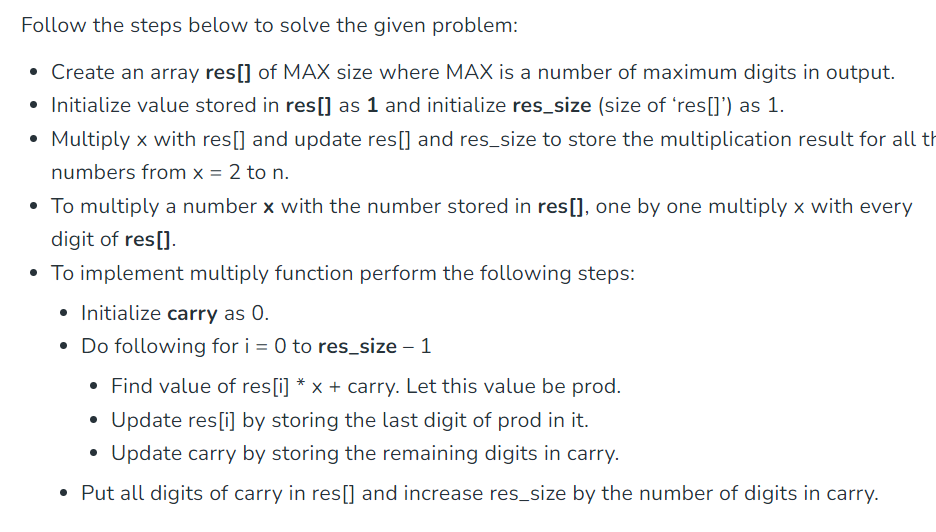
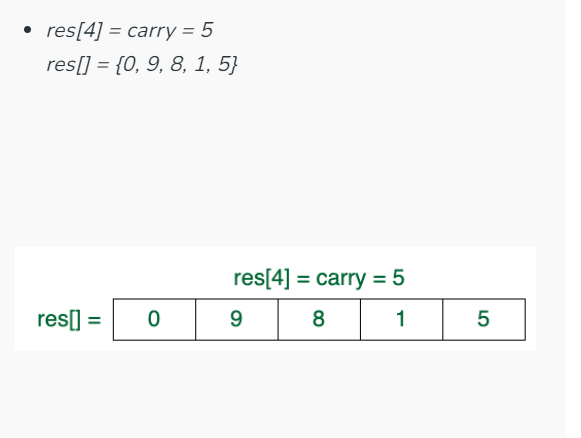
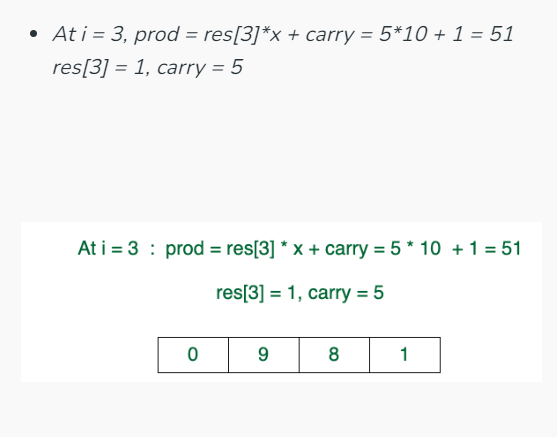
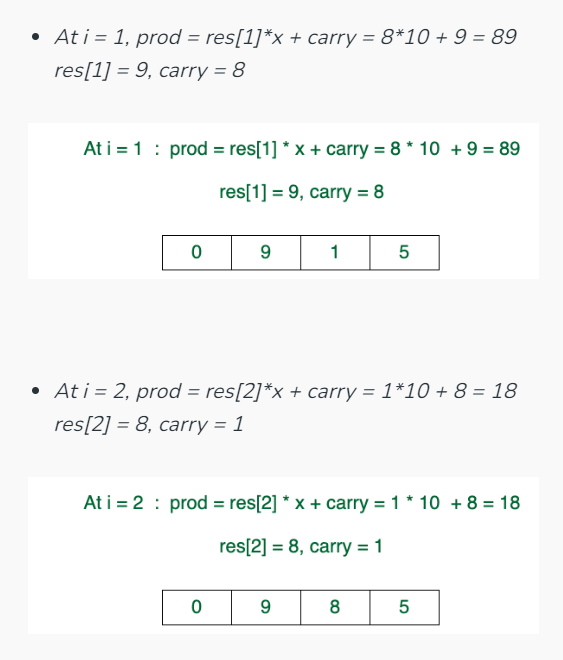
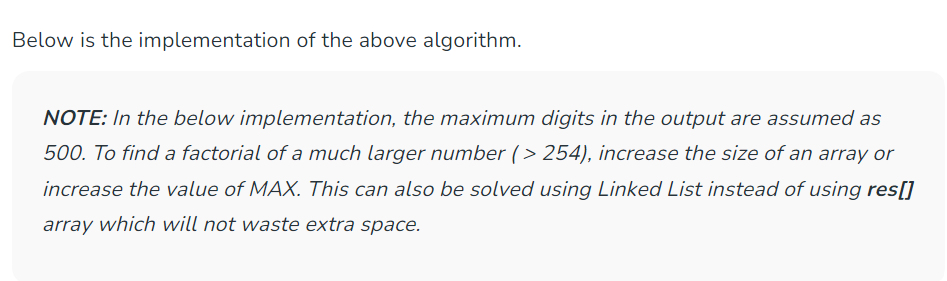
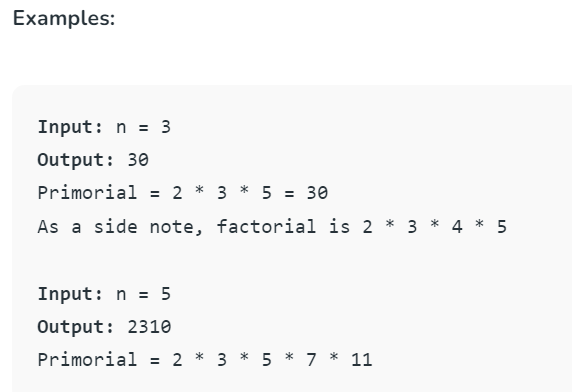
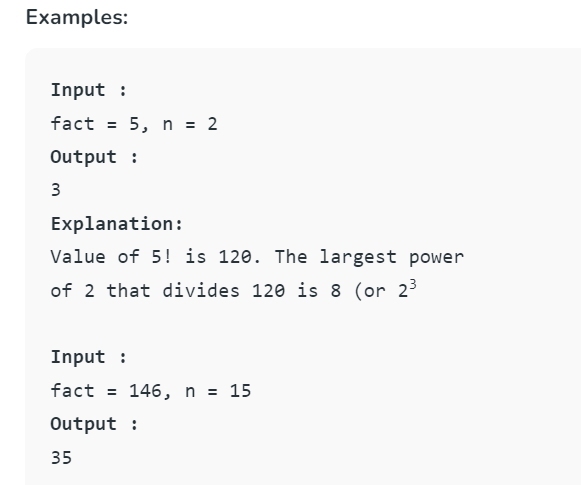
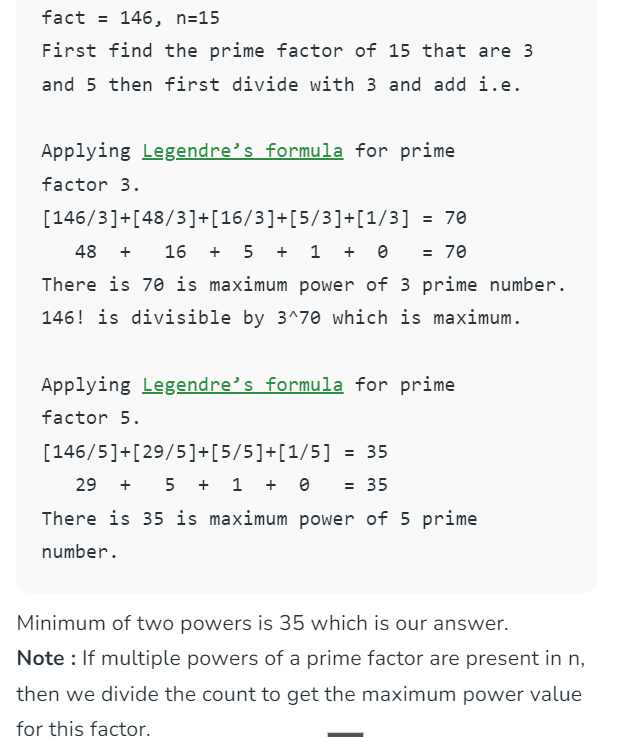
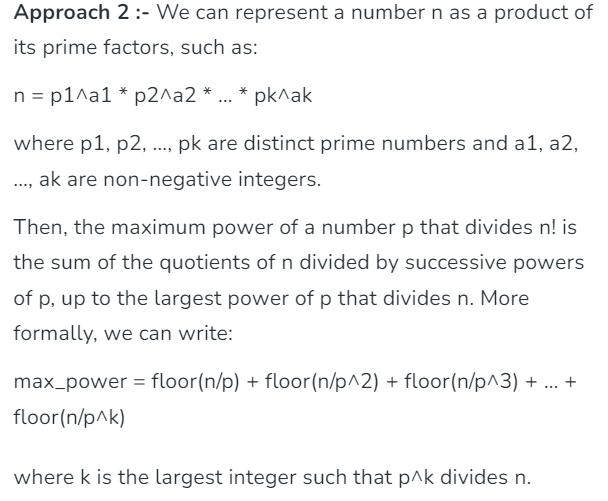
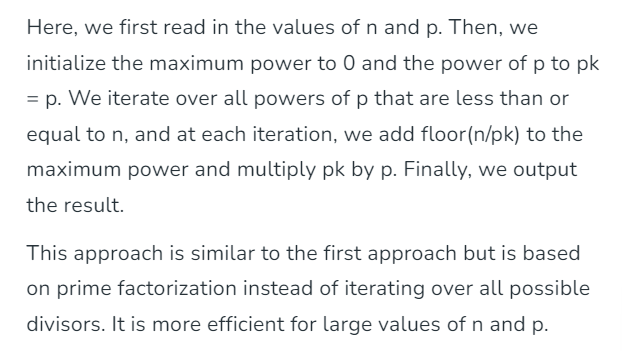
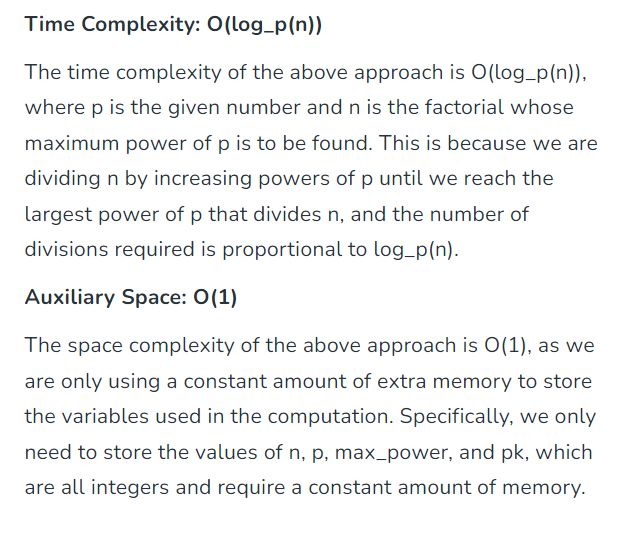
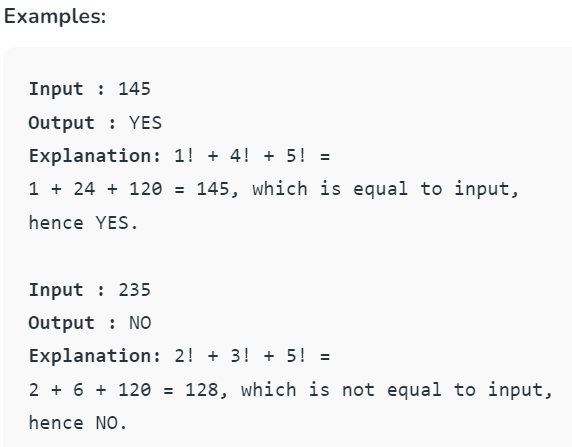
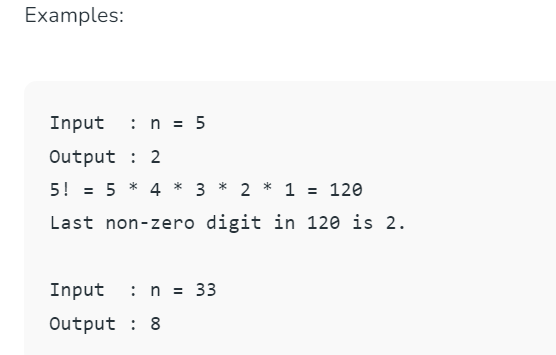
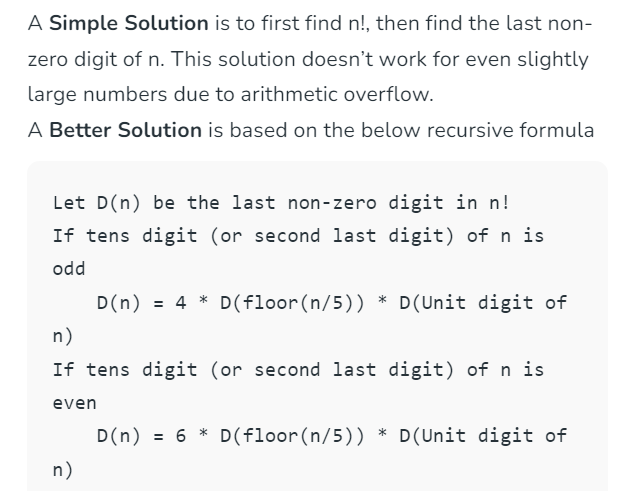
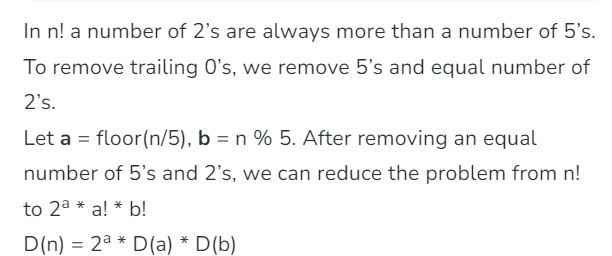
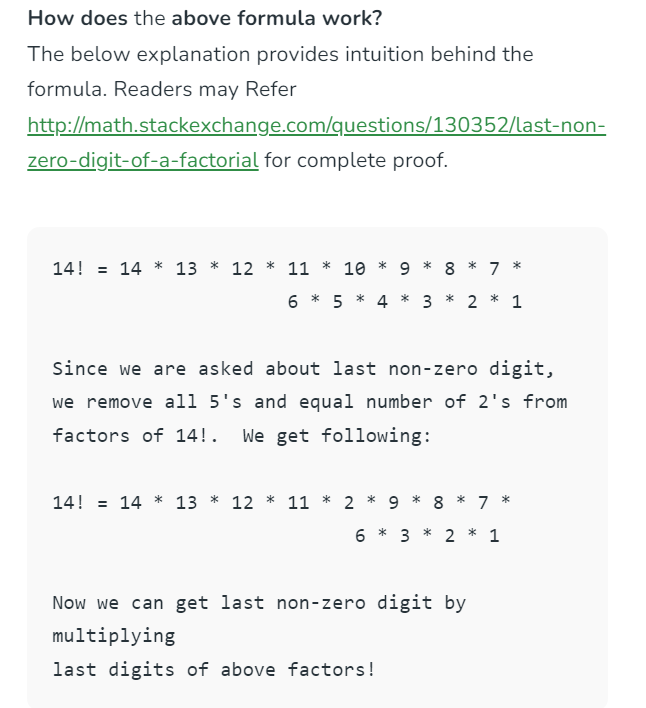
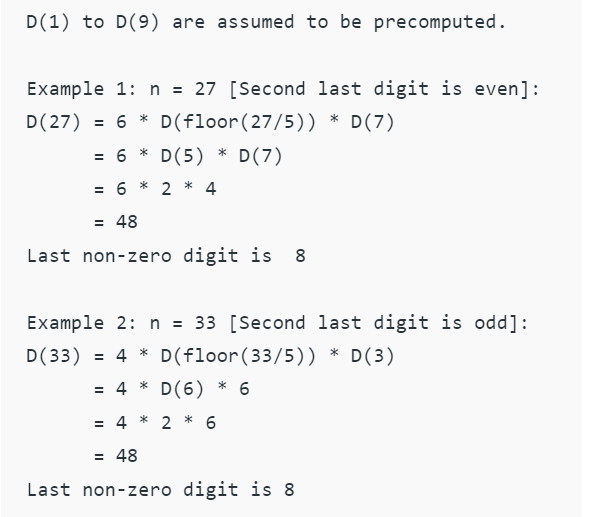
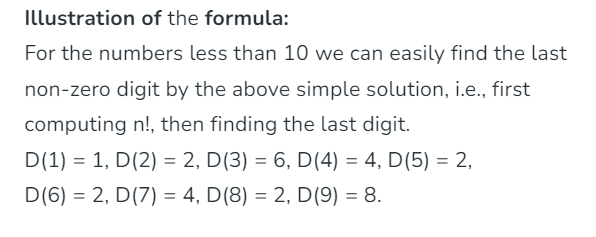
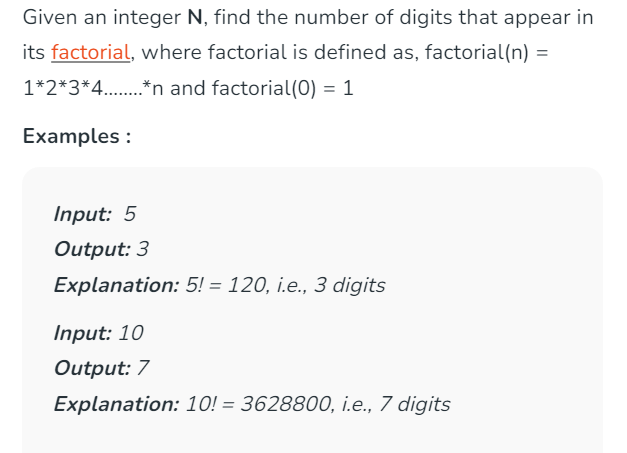
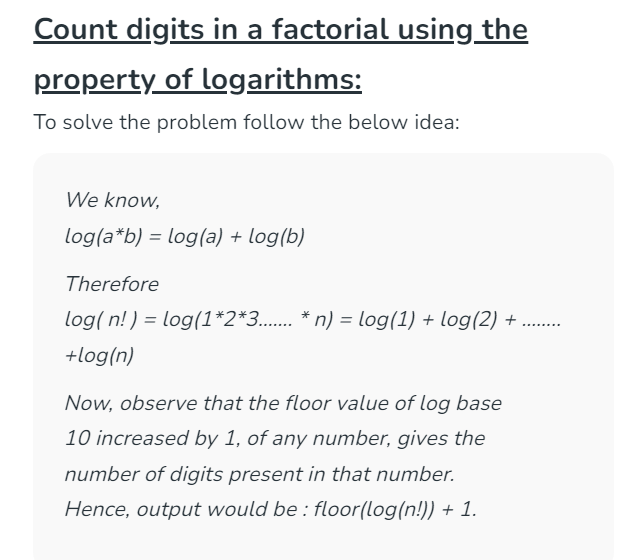
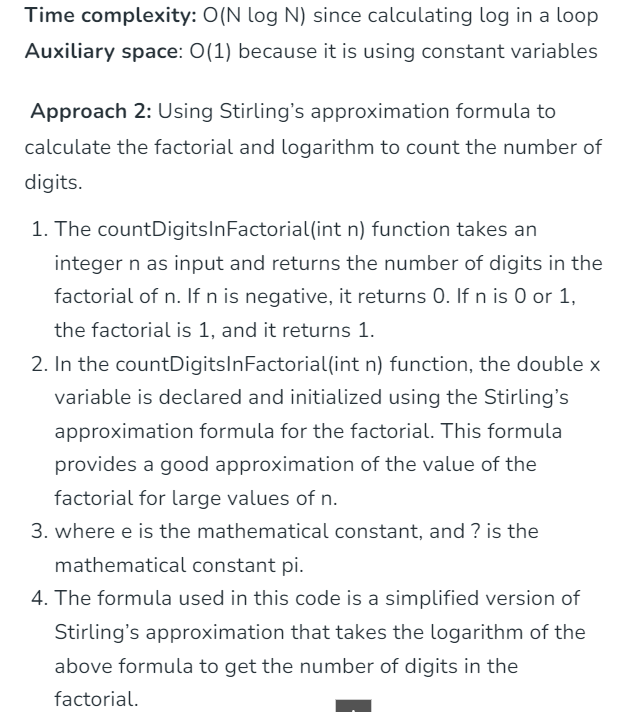
**Factorial**

1. **Factorial of a number.**
   1. **Widely used in permutation and combination to calculate the total possible outcomes.**
   2. **Problem in writing code of factorial.**
      1. When the value of n changes increases by 1, the value of the factorial increases by n. So the variable storing the value of factorial should have a large size. Following is the value of n whose factorial can be stored in the respective size.
      2. integer –> n<=12
      3. long long int –> n<=19
2. **Legendre’s formula**
   1. Given an integer n and a prime number p, find the largest x such that px (p raised to power x) divides n! (factorial) **Factorial of a number.**
      1. ****
      2. ****
      3. ****
      4. ****
   2. What to do if p is not prime.
      1. We can find all prime factors of p and compute result for every prime factor
3. **Trailing zeroes**
   1. Given an integer n, write a function that returns count of trailing zeroes in n!.
   2. ****
   3. ****
   4. ****
   5. ****
   6. ****
   7. **Time Complexity:**O(log5n)  
      **Auxiliary Space:**O(1)
4. **Find Factorial of a large number.**
   1. ****
   2. ****
   3. ****
   4. ****
5. **Primorial of a number.**
   1. Given a number n, the task is to calculate its primorial. Primorial (denoted as Pn#) is a product of first n prime numbers. [Primorial](https://en.wikipedia.org/wiki/Primorial" \t "_blank) of a number is similar to the factorial of a number. In primorial, not all the natural numbers get multiplied only prime numbers are multiplied to calculate the primorial of a number. It is denoted with P#
   2. ****
   3. An **efficient**method is to find all the prime up-to n using [Sieve of Sundaram](https://www.geeksforgeeks.org/sieve-sundaram-print-primes-smaller-n/)and then just calculate the primorial by multiplying them all.
6. **Find maximum power of a number that divides a factorial.**
   1. Given two numbers, **fact** and **n**, find the largest power of n that divides **fact!** (Factorial of fact).
   2. ****
   3. The idea is based on [Legendre’s formula](https://www.geeksforgeeks.org/given-p-and-n-find-the-largest-x-such-that-px-divides-n-2/) which finds largest power of a [prime number](https://www.geeksforgeeks.org/prime-numbers/) that divides fact!. We [find all prime factors of n](https://www.geeksforgeeks.org/print-all-prime-factors-of-a-given-number/). For every prime factor we find largest power of it that divides fact!. Finally we return minimum of all found powers.
   4. ****
   5. **Time Complexity:**O(sqrt(n)\*log(n))  
      **Auxiliary Space:** O(1), as no extra space is used
   6. ****
   7. ****
   8. ****
7. **Largest power pf k in N! where k may not be prime – To DO**
   1. Given two numbers k and n, find the largest power of k that divides n!    
      **Constraints:**  K > 1
8. **Krishnamurthy number.**
   1. A Krishnamurthy number is a number whose sum of the factorial of digits is equal to the number itself.  
      For example, 145 is the sum of the factorial of each digit.  
      **1! + 4! + 5! = 1 + 24 + 120 = 145**
   2. ****
   3. The idea is simple, we compute the sum of factorials of all digits and then compare the sum with n.
   4. **Time Complexity:** O(n log10n) where n is a given number  
      **Auxiliary Space**: O(1)  
      Interestingly, there are exactly four Krishnamurthy numbers i.e. 1, 2, 145, and 40585 known to us.
   5. **Precompute the factorial for digit 0-9**
   6. **Time Complexity: O(logN)**  
      **Auxiliary Space: O(1)**
9. **Last non-zero digit of factorial**
   1. Given a number n, find the last non-zero digit in n!.
   2. ****
   3. ****
   4. ****
   5. **Time complexity:** O(log n)
   6. **Space complexity:** O(log n)
10. **Factorial of a number.**
    1. 
    2. 
    3. 
    4. **Time complexity: O(1)**
    5. The time complexity of the above approach to count the number of digits in n! using Stirling’s approximation and logarithms is O(1), meaning it is constant time complexity.  
       **Auxiliary space: O(1)**