**26. Given a 2D integer array matrix, return the transpose of matrix. The transpose of a matrix is the matrix flipped over its main diagonal, switching the matrix's row and column indices.**

**Example 1: Input: matrix = [[1,2,3],[4,5,6],[7,8,9]] ; Output: [[1,4,7],[2,5,8],[3,6,9]]**

**Example 2: Input: matrix = [[1,2,3],[4,5,6]] ; Output: [[1,4],[2,5],[3,6]]**

**Program:**

def transpose\_matrix(matrix):

return [[matrix[j][i] for j in range(len(matrix))] for i in range(len(matrix[0]))]

matrix1 = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]

print(transpose\_matrix(matrix1)) #

matrix2 = [[1, 2, 3], [4, 5, 6]]

print(transpose\_matrix(matrix2))

**27. Given an integer n, return the nth digit of the infinite integer sequence [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, ...].**

**Example 1: Input: n = 3 Output: 3**

**Example 2: Input: n = 11 Output: 0**

**Explanation: The 11th digit of the sequence 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, ... is a 0, which is part of the number 10**

**Program:**

def findNthDigit(n):

n -= 1

for digits in range(1, 11):

first\_num = 10 \*\* (digits - 1)

if n < 9 \* digits \* first\_num:

return int(str(first\_num + n // digits)[n % digits])

n -= 9 \* digits \* first\_num

return -1 # If n is out of range

# Test the function

print(findNthDigit(3))

print(findNthDigit(11))

**28. Given a sentence that consists of some words separated by a single space, and a searchWord, check if searchWord is a prefix of any word in sentence. Return the index of the word in sentence (1-indexed) where searchWord is a prefix of this word. If searchWord is a prefix of more than one word, return the index of the first word (minimum index). If there is no such word return - 1. A prefix of a string s is any leading contiguous substring of s.**

**Example 1: Input: sentence = "i love eating burger", searchWord = "burg" Output: 4**

**Explanation: "burg" is prefix of "burger" which is the 4th word in the sentence**

**Program:**

def is\_prefix(sentence, searchWord):

words = sentence.split()

for idx, word in enumerate(words, 1):

if word.startswith(searchWord):

return idx

return -1

# Test the function

sentence = "i love eating burger"

searchWord = "burg"

print(is\_prefix(sentence, searchWord))

**29. Given an integer array num sorted in non-decreasing order. You can perform the following operation any number of times: Choose two indices, i and j, where nums[i] < nums[j]. Then, remove the elements at indices i and j from nums. The remaining elements retain their original order, and the array is reindexed. Return the minimum length of nums after applying the operation zero or more times. Example 1: Input: nums = [1,2,3,4] Output: 0**

**Constraints: 1 <= nums.length <= 105 1 <= nums[i] <= 109 nums is sorted in non-decreasing order.**

**Program:**

def min\_length\_after\_removal(nums):

result = len(nums)

for i in range(len(nums) - 1):

if nums[i] < nums[i + 1]:

result -= 2

if len(nums) > 1 and nums[-1] > nums[-2]:

result -= 1

return max(result, 0)

nums1 = [1, 2, 3, 4]

print(min\_length\_after\_removal(nums1))

**30. Given an array of string words, return all strings in words that is a substring of another word. You can return the answer in any order. A substring is a contiguous sequence of characters within a string Example 1: Input: words = ["mass","as","hero","superhero"] ; Output: ["as","hero"]**

**Explanation: "as" is substring of "mass" and "hero" is substring of "superhero". ["hero","as"] is also a valid answer.**

**Program:**

def find\_substrings(words):

substrings = []

for word1 in words:

for word2 in words:

if word1 != word2 and word1 in word2:

substrings.append(word1)

break

return list(set(substrings))

words = ["mass", "as", "hero", "superhero"]

print(find\_substrings(words))

# Output: ['as', 'hero']