**Question 1:**

Problem: Given an array of integers, find the length of the longest subsequence of elements that are in increasing order.

Example: Input:

    [10, 9, 2, 5, 3, 7, 101, 18]

     Output: 4 Explanation: The longest increasing subsequence is [2, 3, 7, 101], which has a length of 4.

**Hint:**

for (int i = 1; i < n; i++) {  
for (int j = 0; j < i; j++) {  
if (nums[i] > nums[j]) {  
dp[i] = Math.max(dp[i], dp[j] + 1);  
maxLen = Math.max(maxLen, dp[i]);  
}  
}  
}

public class Main {

    public static void main(String[] args) {

        int[] arr = {10, 9, 2, 5, 3, 7, 101, 18};

        find(arr);

    }

    public static void find(int[] arr) {

        int maxCount = 0;

        for (int i = 0; i < arr.length; i++) {

            int internalCount = 0;

            for (int j = 0 + i; j < arr.length; j++) {

                if (arr[i] > arr[j]) {

                    internalCount++;

                }

            }

            if (maxCount < internalCount) {

                maxCount = internalCount;

            }

        }

        System.out.println(maxCount);

    }

}

**Question 2:**

Here's a classic dynamic programming problem called the "Coin Change Problem." It's similar in difficulty to the "Longest Increasing Subsequence" problem, and it involves finding the number of ways to make change for a given target amount using a set of coin denominations.

Coin Change Problem:

You are given an array of coin denominations and a total amount. Determine the number of distinct ways you can make change for the given amount using the provided coins. Assume you have an unlimited supply of each coin denomination.

Example:

Given coins: [1, 2, 5] Amount: 5

Output: 4

Explanation: There are four ways to make change for the amount 5 using the coin denominations [1, 2, 5]: [1, 1, 1, 1, 1], [1, 1, 1, 2], [2, 2, 1], and [5]

**Hint:**

dp[0] = 1; // Base case  
  
for (int coin : coins) {  
for (int i = coin; i <= amount; i++) {  
dp[i] += dp[i - coin];  
}  
}  
  
return dp[amount];

**Question 3:**

"3Sum" problem:

Given an array of integers, find all unique triplets in the array that sum up to a target value.

Example:

Input: nums = [-1, 0, 1, 2, -1, -4], target = 0 Output: [[-1, -1, 2], [-1, 0, 1]]

**Question 4:**

Longest Common Subsequence (LCS):

Given two strings text1 and text2, find the length of their longest common subsequence.

A subsequence is a sequence that appears in the same order, but not necessarily consecutively, in both strings. If there is no common subsequence, return 0.

Example:

Input: text1 = "abcde", text2 = "ace" Output: 3 Explanation: The longest common subsequence is "ace", which has a length of 3.

**Question 5:**

Knapsack Problem:

You are given a set of items, each with a weight and a value, and a knapsack with a maximum weight capacity. Your goal is to maximize the total value of the items you can include in the knapsack without exceeding its weight capacity.

Example:

Input:

* Values: [60, 100, 120]
* Weights: [10, 20, 30]
* Knapsack Capacity: 50

Output: 220 Explanation: The optimal choice is to take items with weights 20 and 30 for a total value of 100 + 120 = 220.

Input:

* Values: [30, 40, 50, 60]
* Weights: [10, 20, 30, 40]
* Knapsack Capacity: 60

Output: 100 Explanation: The optimal choice is to take items with weights 20 and 40 for a total value of 40 + 60 = 100.

**Assignment:**

Post Entity

1. Post ID (String) PK

2. Post Title

3. Post Content

4. Author

5. Date Time Posted