Day1 programs:

Q1: Write a Program that accepts two Strings as command line arguments and generate the output in the required format.

Example1)

If the two command line arguments are Wipro and Bangalore then the output generated should be Wipro Technologies Bangalore.

public class Main

{

public static void main(String[] args) {

String a="technologies";

System.out.println(args[0]+" "+a+" "+args[1]);

}

}

Q2: Write a Program to accept a String as a command line argument and print a Welcome message as given below.

Example1)

C:\> java Sample John

O/P Expected : Welcome John

public class Main

{

public static void main(String[] args) {

String a="john";

System.out.println(args[0]+" "+a);

}

}

Write a Program to accept a String as a command line argument and print a Welcome message as given below.

Example1)

C:\> java Sample John

O/P Expected : Welcome John

public class Main

{

public static void main(String[] args)

{

int a=Integer.parseInt(args[0]);

int b=Integer.parseInt(args[1]);

int sum=a+b;

System.out.println("the sum of two numbers is:"+sum);

}

}

Here are some common coding problems that are frequently asked in technical interviews, organized by category:

**1. Arrays and Strings**

* **Two Sum**: Given an array of integers, return indices of the two numbers such that they add up to a specific target.
* **Maximum Subarray**: Find the contiguous subarray within a one-dimensional array of numbers that has the largest sum (Kadane's Algorithm).
* **Rotate Array**: Rotate the array to the right by k steps, where k is non-negative.
* **Merge Intervals**: Given a collection of intervals, merge all overlapping intervals.
* **Trapping Rain Water**: Given n non-negative integers representing an elevation map where the width of each bar is 1, compute how much water it can trap after raining.

**2. Linked Lists**

* **Reverse a Linked List**: Reverse a singly linked list.
* **Detect Cycle in a Linked List**: Given a linked list, determine if it has a cycle in it.
* **Merge Two Sorted Lists**: Merge two sorted linked lists and return it as a new sorted list.
* **Remove Nth Node From End of List**: Remove the nth node from the end of a linked list.
* **Linked List Cycle II**: Return the node where the cycle begins in a linked list.

**3. Trees and Graphs**

* **Binary Tree Inorder Traversal**: Given a binary tree, return the inorder traversal of its nodes' values.
* **Lowest Common Ancestor of a Binary Search Tree**: Given a binary search tree, find the lowest common ancestor of two given nodes.
* **Level Order Traversal**: Given a binary tree, return the level order traversal of its nodes' values.
* **Binary Tree Maximum Path Sum**: Find the maximum path sum in a binary tree.
* **Clone Graph**: Clone an undirected graph where each node contains a label and a list of its neighbors.

**4. Dynamic Programming**

* **Climbing Stairs**: You are climbing a staircase. It takes n steps to reach the top. Each time you can either climb 1 or 2 steps. How many distinct ways can you climb to the top?
* **Longest Increasing Subsequence**: Find the length of the longest increasing subsequence in an array.
* **Coin Change**: Given an integer array coins and an integer amount, return the fewest number of coins that you need to make up that amount.
* **0/1 Knapsack Problem**: Given weights and values of n items, put these items in a knapsack of capacity W to get the maximum total value in the knapsack.
* **Edit Distance**: Given two words word1 and word2, find the minimum number of operations required to convert word1 to word2.

**5. Backtracking**

* **Permutations**: Given a collection of distinct integers, return all possible permutations.
* **Combinations**: Given two integers n and k, return all possible combinations of k numbers out of 1 ... n.
* **N-Queens Problem**: Place n queens on an n x n chessboard so that no two queens threaten each other.
* **Sudoku Solver**: Write a program to solve a Sudoku puzzle by filling the empty cells.
* **Subsets**: Return all possible subsets of a set of integers.

**6. Sorting and Searching**

* **Merge Sort**: Implement the merge sort algorithm.
* **Quick Sort**: Implement the quick sort algorithm.
* **Binary Search**: Implement binary search on a sorted array.
* **Find Peak Element**: A peak element is an element that is strictly greater than its neighbors. Find a peak element in the array.
* **Search in Rotated Sorted Array**: Suppose an array is rotated at some pivot unknown to you beforehand. Search for a given target value.

**7. String Manipulation**

* **Longest Palindromic Substring**: Find the longest palindromic substring in a given string.
* **String to Integer (atoi)**: Implement the function atoi, which converts a string to an integer.
* **Group Anagrams**: Given an array of strings, group anagrams together.
* **Valid Parentheses**: Given a string containing just the characters '(', ')', '{', '}', '[' and ']', determine if the input string is valid.
* **Longest Common Prefix**: Write a function to find the longest common prefix string amongst an array of strings.