**Softnerve Tech Assement**

1. **8/10**

Arrays, Linked Lists, Stack, Queue, Recursion, Trees, Graphs, Dynamic Programming, Greedy Algorithms, Bit Manipulation, Heaps, Divide and Conquer

1. **HackerRank**

[**https://www.hackerrank.com/themanisha017?hr\_r=1**](https://www.hackerrank.com/themanisha017?hr_r=1)

1. **7/10**
2. **NA**

**Question 1: Leader in the Array**

import java.util.ArrayList;

import java.util.List;

public class LeaderInArray {

    public static List<Integer> findLeaders(int[] arr) {

        List<Integer> leaders = new ArrayList<>();

        int n = arr.length;

        int maxElement = arr[n - 1];

        leaders.add(maxElement);

        for (int i = n - 2; i >= 0; i--) {

            if (arr[i] > maxElement) {

                maxElement = arr[i];

                leaders.add(maxElement);

            }

        }

        return leaders;

    }

    public static void main(String[] args) {

        int[] arr = { 7, 10, 4, 10, 6, 5, 2 };

        List<Integer> leaders = findLeaders(arr);

        System.out.println("Leaders in the array: " + leaders);

    }

}

**Question 2: Best Time to Buy and Sell Stock**

public class BuySellStock {

    public static int maxProfit(int[] prices) {

        int minPrice = Integer.MAX\_VALUE;

        int maxProfit = 0;

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

            if (prices[i] < minPrice) {

                minPrice = prices[i];

            } else if (prices[i] - minPrice > maxProfit) {

                maxProfit = prices[i] - minPrice;

            }

        }

        return maxProfit;

    }

    public static void main(String[] args) {

        int[] prices1 = { 7, 1, 5, 3, 6, 4 };

        int maxProfit1 = maxProfit(prices1);

        System.out.println("Max Profit: " + maxProfit1);

        int[] prices2 = { 7, 6, 4, 3, 1 };

        int maxProfit2 = maxProfit(prices2);

        System.out.println("Max Profit: " + maxProfit2);

    }

}

**Question 3: Sum of All Subset XOR Totals**

public class SubsetXORTotals {

    public static int subsetXORSum(int[] nums) {

        return calculateXORTotals(nums, 0, 0);

    }

    public static int calculateXORTotals(int[] nums, int index, int currentXOR) {

        if (index == nums.length) {

            return currentXOR;

        }

        int includeXOR = calculateXORTotals(nums, index + 1, currentXOR ^ nums[index]);

        int excludeXOR = calculateXORTotals(nums, index + 1, currentXOR);

        return includeXOR + excludeXOR;

    }

    public static void main(String[] args) {

        int[] nums1 = { 1, 3 };

        int xorSum1 = subsetXORSum(nums1);

        System.out.println("XOR Sum: " + xorSum1);

        int[] nums2 = { 5, 1, 6 };

        int xorSum2 = subsetXORSum(nums2);

        System.out.println("XOR Sum: " + xorSum2);

    }

}