Basic Questions (Easy Level)

1. Sum of Array Elements

Problem Statement:
Write a Java program that takes an array of integers and returns the sum of all elements.

Test Cases:

```
Input: [1, 2, 3, 4, 5]
Output: 15

Input: [10, 20, 30]
Output: 60

Input: [-5, 5, -10, 10]
Output: 0
```

Explanation:
Traverse the array and sum all elements.

---

2. Find the Maximum Element in an Array

Problem Statement:
Find the largest number in a given array.

Test Cases:

```
Input: [10, 20, 30, 40, 50]
Output: 50

Input: [-5, -10, -1, -20]
Output: -1

Input: [99, 75, 100, 120, 110]
Output: 120
```

Explanation:
Iterate through the array and track the maximum value.

---

3. Find the Minimum Element in an Array

Problem Statement:
Find the smallest number in a given array.

Test Cases:

```
Input: [10, 20, 30, 40, 50]
Output: 10

Input: [-5, -10, -1, -20]
Output: -20

Input: [99, 75, 100, 120, 110]
Output: 75
```

Explanation:
Iterate through the array and track the minimum value.

---

4. Reverse an Array

Problem Statement:
Reverse the order of elements in an array.

Test Cases:

```
Input: [1, 2, 3, 4, 5]
Output: [5, 4, 3, 2, 1]

Input: [10, 20, 30, 40]
Output: [40, 30, 20, 10]

Input: [99]
Output: [99]
```

Explanation:
Swap elements from start to end.

---

5. Check if an Array Contains a Specific Element

Problem Statement:
Check if a target number exists in an array.

Test Cases:

```
Input: [1, 2, 3, 4, 5], Target: 3
Output: true

Input: [10, 20, 30, 40], Target: 50
Output: false

Input: [99, 75, 100, 120], Target: 120
Output: true
```

Explanation:
Loop through the array and check if the number exists.

---

6. Count Occurrences of an Element in an Array

Problem Statement:
Count how many times a given number appears in an array.

Test Cases:

```
Input: [1, 2, 2, 3, 3, 3, 4], Target: 3
Output: 3

Input: [10, 10, 20, 30, 10], Target: 10
Output: 3

Input: [99, 75, 100, 120, 99], Target: 99
Output: 2
```

Explanation:
Use a loop and count occurrences.

---

7. Copy Elements of One Array to Another

Problem Statement:
Copy all elements from one array into another.

Test Cases:

```
Input: [1, 2, 3, 4, 5]
Output: [1, 2, 3, 4, 5]

Input: [10, 20, 30]
Output: [10, 20, 30]

Input: [99, 75, 100, 120]
Output: [99, 75, 100, 120]
```

Explanation:
Create a new array and copy values.

---

8. Find the Average of Array Elements

Problem Statement:
Compute the average (mean) of numbers in an array.

Test Cases:

```
Input: [1, 2, 3, 4, 5]
Output: 3.0

Input: [10, 20, 30]
Output: 20.0

Input: [99, 75, 100, 120]
```

```
Output: 98.5
```

Explanation:
Sum all elements and divide by the number of elements.

---

9. Merge Two Arrays

Problem Statement:
Merge two arrays into a single array.

Test Cases:

```
Input: [1, 2, 3], [4, 5, 6]
Output: [1, 2, 3, 4, 5, 6]

Input: [10, 20], [30, 40, 50]
Output: [10, 20, 30, 40, 50]

Input: [99], [75, 100, 120]
Output: [99, 75, 100, 120]
```

Explanation:
Create a new array and add elements from both.

---

10. Find the Index of an Element in an Array

Problem Statement:
Return the index of a given number in an array, or `-1` if not found.

Test Cases

```
Input: [1, 2, 3, 4, 5], Target: 3
Output: 2

Input: [10, 20, 30, 40], Target: 50
Output: -1
```

```
Input: [99, 75, 100, 120], Target: 120
Output: 3
```

Explanation:
Loop through the array and return the index when found.

Intermediate Questions (Medium Level)

11. Find the Second Largest Element in an Array

Problem Statement:
Find the second largest number in an **unsorted array** of unique integers.

```
Input: [10, 5, 8, 20]
Output: 10

Input: [1, 2, 3, 4, 5]
Output: 4

Input: [99, 75, 100, 120, 110]
Output: 110

Input: [5]
Output: "Not enough elements"

Input: [8, 8, 8]
Output: "Not enough distinct elements"
```

Explanation:

1. Traverse the array while keeping track of the **largest** and **second largest** numbers.
2. If there is **only one unique element**, return an error message.

---

12. Find the Second Smallest Element in an Array

Problem Statement:
Find the second smallest number in an **unsorted array** of unique integers.

Test Cases:
```
Input: [10, 5, 8, 20]
```

```
Output: 8

Input: [1, 2, 3, 4, 5]
Output: 2

Input: [99, 75, 100, 120, 110]
Output: 99

Input: [3]
Output: "Not enough elements"

Input: [7, 7, 7, 7]
Output: "Not enough distinct elements"
```

Explanation:

1. Traverse the array while keeping track of the **smallest** and **second smallest** numbers.
2. If there is **only one unique element**, return an error message.

---

13. Remove Duplicates from an ArrayList

Problem Statement:
Given an `ArrayList<Integer>`, remove all **duplicate elements** and return a list of unique elements.

Test Cases:

```
Input: [1, 2, 2, 3, 4, 4, 5]
Output: [1, 2, 3, 4, 5]

Input: [10, 20, 10, 30, 40, 30]
Output: [10, 20, 30, 40]

Input: [5, 5, 5, 5, 5]
Output: [5]

Input: []
Output: []
```

Explanation:
Use a `HashSet` to filter out duplicates.

---

14. Sort an Array Without Using Built-in Methods

Problem Statement:
Sort an array without using Java's built-in sorting functions (`Arrays.sort()`).

Test Cases:

```
Input: [4, 2, 9, 1, 5]
Output: [1, 2, 4, 5, 9]

Input: [99, 75, 100, 120, 110]
Output: [75, 99, 100, 110, 120]

Input: [3, 3, 3, 3]
Output: [3, 3, 3, 3]
```

Explanation:
Implement **Bubble Sort, Selection Sort, or Insertion Sort** to sort the array.

---

15. Find the Common Elements Between Two Arrays

Problem Statement:
Return an array containing the **common elements** in two arrays.

Test Cases:

```
Input: [1, 2, 3, 4, 5], [3, 4, 5, 6, 7]

Output: [3, 4, 5]

Input: [10, 20, 30, 40], [50, 60, 70]
Output: []
```

```
Input: [99, 75, 100, 120], [75, 100, 130, 140]
Output: [75, 100]
```

Explanation:

1. Use a `HashSet` to store elements from one array.
2. Check if elements of the second array exist in the set.

---

16. Shift Array Elements to the Left by One Position

Problem Statement:
Shift the array elements to the **left** by one position.

Test Cases:

```
Input: [1, 2, 3, 4]
Output: [2, 3, 4, 1]

Input: [10, 20, 30]
Output: [20, 30, 10]

Input: [5]
Output: [5]
```

Explanation:

1. Store the first element.
2. Move all elements one step left.
3. Place the first element at the end.

---

17. Shift Array Elements to the Right by One Position

Problem Statement:
Shift the array elements to the **right** by one position.

Test Cases:

```
Input: [1, 2, 3, 4]
Output: [4, 1, 2, 3]

Input: [10, 20, 30]
Output: [30, 10, 20]

Input: [5]
Output: [5]
```

Explanation:

1. Store the last element.
2. Move all elements one step right.
3. Place the last element at the beginning.

---

18. Find Pairs with a Given Sum in an Array

Problem Statement:
Find all pairs of numbers that add up to a **target sum**.

Test Cases:

```
Input: [1, 2, 3, 4, 5], Target: 5
Output: [(1, 4), (2, 3)]

Input: [10, 20, 30, 40, 50], Target: 60
Output: [(10, 50), (20, 40)]

Input: [5, 5, 5, 5], Target: 10
Output: [(5, 5), (5, 5)]
```

Explanation:
Use **nested loops or HashMap** to find pairs efficiently.

---

19. Check if Two Arrays are Equal

Problem Statement:
Return `true` if both arrays have the **same elements** in the **same order**.

Test Cases:

```
Input: [1, 2, 3], [1, 2, 3]
Output: true

Input: [1, 2, 3], [3, 2, 1]
Output: false

Input: [5, 5, 5], [5, 5, 5]
Output: true
```

Explanation:

1. Compare **lengths** of both arrays.
2. Compare **each element** one by one.

---

20. Find the Intersection of Two Arrays

Problem Statement:
Find the **common elements** between two arrays, maintaining unique values.

Test Cases:

```
Input: [1, 2, 3, 4, 5], [3, 4, 5, 6, 7]
Output: [3, 4, 5]

Input: [10, 20, 30, 40], [50, 60, 70]
Output: []

Input: [99, 75, 100, 120], [75, 100, 130, 140]
Output: [75, 100]
```

Explanation:
Use **two HashSets** to store elements and find common values.

Advanced Questions(Hard Level)

21. Rotate an Array by K Positions (Right Rotation)

Problem Statement:
Rotate an array to the **right** by `k` positions.

Test Cases:

```
Input: [1, 2, 3, 4, 5], k = 2

Output: [4, 5, 1, 2, 3]

Input: [10, 20, 30, 40, 50], k = 3

Output: [30, 40, 50, 10, 20]

Input: [99, 75, 100, 120], k = 1

Output: [120, 99, 75, 100]
```

Explanation:

1. Reverse the entire array.
2. Reverse the first `k` elements.
3. Reverse the remaining elements.

---

22. Rotate an Array by K Positions (Left Rotation)

Problem Statement:
Rotate an array to the **left** by `k` positions.

Test Cases:

```
Input: [1, 2, 3, 4, 5], k = 2

Output: [3, 4, 5, 1, 2]

Input: [10, 20, 30, 40, 50], k = 3
```

```
Output: [40, 50, 10, 20, 30]

Input: [99, 75, 100, 120], k = 1

Output: [75, 100, 120, 99]
```

Explanation:

1. Reverse the first `k` elements.
2. Reverse the remaining elements.
3. Reverse the entire array.

---

23. Find Missing Number in a Consecutive Series

Problem Statement:
Find the missing number from a given range `[1, N]`.

Test Cases:

makefile

CopyEdit

```
Input: [1, 2, 4, 5], N = 5

Output: 3

Input: [3, 7, 1, 2, 8, 4, 5], N = 8

Output: 6

Input: [10, 11, 12, 14], N = 14

Output: 13
```

Explanation:

1. Use the **sum formula** `N * (N + 1) / 2` and subtract the sum of elements in the array.

24. Find the Subarray with Maximum Sum (Kadane's Algorithm)

Problem Statement:
Find the **maximum sum subarray** from a given array.

Test Cases:

```
Input: [-2, 1, -3, 4, -1, 2, 1, -5, 4]

Output: 6  (Subarray: [4, -1, 2, 1])

Input: [1, 2, 3, -2, 5]

Output: 9  (Subarray: [1, 2, 3, -2, 5])

Input: [-1, -2, -3, -4]

Output: -1  (Subarray: [-1])
```

Explanation:
Use **Kadane's Algorithm** to find the max subarray sum in O(N) time.

25. Find All Triplets in an Array That Sum to Zero

Problem Statement:
Find all unique triplets `(a, b, c)` where `a + b + c = 0`.

Test Cases:

```
Input: [-1, 0, 1, 2, -1, -4]

Output: [[-1, -1, 2], [-1, 0, 1]]

Input: [0, 1, 1]

Output: []
```

```
Input: [0, -1, 2, -3, 1]

Output: [[-3, 1, 2], [-1, 0, 1]]
```

Explanation:

1. Sort the array.
2. Use a **two-pointer approach** to find valid triplets.

---

26. Rearrange Array in Alternating Positive and Negative Order

Problem Statement:
Rearrange the array such that positive and negative numbers alternate.

Test Cases:

```
Input: [1, 2, -3, -4, 5, -6]

Output: [1, -3, 2, -4, 5, -6]

Input: [-5, -2, 5, 2, 4, 7, 1, 8, 0, -8]

Output: [5, -5, 2, -2, 4, -8, 7, 1, 8, 0]

Input: [1, 2, 3, 4, 5]

Output: [1, 2, 3, 4, 5]
```

Explanation:
Use two separate lists and merge them while alternating elements.

---

27. Find the Majority Element (Element Occurring More Than N/2 Times)

Problem Statement:
Find the element that appears more than `N/2` times in an array.

Test Cases:

```
Input: [3, 3, 4, 2, 4, 4, 2, 4, 4]

Output: 4

Input: [1, 2, 3, 3, 3, 3, 3]

Output: 3

Input: [1, 1, 2, 2, 3, 3]

Output: "No Majority Element"
```

Explanation:
Use **Boyer-Moore Voting Algorithm** to find the majority element in O(N) time.

---

28. Find the Longest Consecutive Sequence in an Array

Problem Statement:
Find the length of the longest consecutive sequence in an unsorted array.

Test Cases:

```
Input: [100, 4, 200, 1, 3, 2]

Output: 4  (Sequence: [1, 2, 3, 4])

Input: [0, 3, 7, 2, 5, 8, 4, 6, 1, 9]

Output: 10  (Sequence: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9])

Input: [1, 9, 3, 10, 4, 20, 2]

Output: 4  (Sequence: [1, 2, 3, 4])
```

Explanation:
Use a **HashSet** to store numbers and check consecutive sequences efficiently.

---

29. Find the First Missing Positive Integer

Problem Statement:
Find the smallest missing positive integer from an unsorted array.

Test Cases:

```
Input: [3, 4, -1, 1]

Output: 2

Input: [1, 2, 0]

Output: 3

Input: [7, 8, 9, 11, 12]

Output: 1
```

Explanation:
Use the **Cyclic Sort Algorithm** to place numbers in correct positions.

---

30. Find the Largest Rectangle in a Histogram (Hard)

Problem Statement:
Given an array representing the heights of bars in a histogram, find the area of the largest rectangle.

Test Cases:

```
Input: [2, 1, 5, 6, 2, 3]

Output: 10

Input: [2, 4]

Output: 4

Input: [6, 2, 5, 4, 5, 1, 6]

Output: 12
```

Explanation:
Use **Stack-based Monotonic approach** for an O(N) solution.