1)Given an array of size N containing only 0s, 1s, and 2s; sort the array in ascending order.

Example 1:

Input:
N = 5
arr[]= {0 2 1 2 0}
Output:
0 0 1 2 2
Explanation:
0s 1s and 2s are segregated
into ascending order.
Example 2:

Input:
N = 3
arr[] = {0 1 0}
Output:
0 0 1
Explanation:
0s 1s and 2s are segregated
into ascending order.

Your Task:
You don't need to read input or print anything. Your task is to complete the function sort012() that takes an array arr and N as input parameters and sorts the array in-place.

Expected Time Complexity: O(N)
Expected Auxiliary Space: O(1)

Ans:

```java
public class Solution {
    public static void sort012(int a[], int n) {
        int low = 0, mid = 0, high = n - 1;
        while (mid <= high) {
            switch (a[mid]) {
                case 0:
                    swap(a, low++, mid++);
                    break;
                case 1:
                    mid++;
                    break;
                case 2:
                    swap(a, mid, high--);
                    break;
            }
        }
    }

    public static void swap(int[] arr, int i, int j) {
        int temp = arr[i];
        arr[i] = arr[j];
        arr[j] = temp;
    }

    public static void main(String[] args) {
        int[] arr1 = {0, 2, 1, 2, 0};
        int n1 = arr1.length;
        sort012(arr1, n1);
        System.out.println("Sorted array 1:");
        for (int num : arr1) {
            System.out.print(num + " ");
        }
        System.out.println();

        int[] arr2 = {0, 1, 0};
        int n2 = arr2.length;
        sort012(arr2, n2);
        System.out.println("Sorted array 2:");
        for (int num : arr2) {
            System.out.print(num + " ");
        }
        System.out.println();
    }
}
```

2)Given an array of size N-1 such that it only contains distinct integers in the range of 1 to N. Find the missing element.

Example 1:
Input:
N = 5
A[] = {1,2,3,5}
Output: 4

Example 2:
Input:
N = 10
A[] = {6,1,2,8,3,4,7,10,5}
Output: 9

Your Task :
You don't need to read input or print anything. Complete the function MissingNumber() that takes array and N as input parameters and returns the value of the missing number.

Expected Time Complexity: O(N)
Expected Auxiliary Space: O(1)

Constraints:
$1 \leq N \leq 106$
$1 \leq A[i] \leq 106$

NOTE: Companies
AMAZON, FLIPKART,ADOBE, CISCO, OLA CABS

Ans:

```java
public class Solution {
   public static int MissingNumber(int[] array, int n) {
       int total = (n * (n + 1)) / 2;

       int sum = 0;
       for (int num : array) {
           sum += num;
```

```
        }

        return total - sum;
    }

    public static void main(String[] args) {
        int[] arr1 = {1, 2, 3, 5};
        int n1 = 5;
        System.out.println("Missing number for array 1: " + MissingNumber(arr1,
n1));

        int[] arr2 = {6, 1, 2, 8, 3, 4, 7, 10, 5};
        int n2 = 10;
        System.out.println("Missing number for array 2: " + MissingNumber(arr2,
n2));
    }
}
```

3)Given an array A of N elements. Find the majority element in the array. A majority element in an array A of size N is an element that appears strictly more than N/2 times in the array.

Example 1:

Input:
N = 3
A[] = {1,2,3}
Output:
-1
Explanation:
Since, each element in
{1,2,3} appears only once so there
is no majority element.
Example 2:

Input:
N = 5
A[] = {3,1,3,3,2}

Output:
3
Explanation:
Since, 3 is present more
than N/2 times, so it is
the majority element.

Your Task:
The task is to complete the function majorityElement() which returns the majority element in the array. If no majority exists, return -1.

Expected Time Complexity: O(N).
Expected Auxiliary Space: O(1).

Constraints:
$1 \leq N \leq 107$
$0 \leq Ai \leq 106$

NOTE: COMPANIES :
Flipkart, Accorlite, Amazon, Microsoft, D-E-Shaw, Google, Nagarro, Atlassian

Ans:

```java
public class Solution {
   public static int majorityElement(int[] nums) {
       int candidate = -1;
       int count = 0;

       for (int num : nums) {
           if (count == 0) {
               candidate = num;
           }
           count += (num == candidate) ? 1 : -1;
       }

       count = 0;
       for (int num : nums) {
```

```
            if (num == candidate) {
                count++;
            }
        }

        return (count > nums.length / 2) ? candidate : -1;
    }

    public static void main(String[] args) {
        int[] arr1 = {1, 2, 3};
        System.out.println("Majority element for array 1: " +
majorityElement(arr1));

        int[] arr2 = {3, 1, 3, 3, 2};
        System.out.println("Majority element for array 2: " +
majorityElement(arr2));
    }
}
```

4)Given two arrays a[] and b[] of size n and m respectively. The task is to find the number of elements in the union between these two arrays.

The Union of the two arrays can be defined as the set containing distinct elements from both arrays. If there are repetitions, then only one occurrence of an element should be printed in the union.

Note : Elements are not necessarily distinct.

Example 1:

Input:
5 3
1 2 3 4 5
1 2 3
Output:
5
Explanation:
1, 2, 3, 4 and 5 are the
elements which comes in the union set

of both arrays. So count is 5.
Example 2:

Input:
6 2
85 25 1 32 54 6
85 2
Output:
7
Explanation:
85, 25, 1, 32, 54, 6, and
2 are the elements which comes in the
union set of both arrays. So count is 7.
Your Task:
Complete the doUnion function that takes a, n, b, m as parameters and returns the count of union elements of the two arrays. The printing is done by the driver code.

Constraints:
$1 \leq n, m \leq 105$
$0 \leq a[i], b[i] < 105$

Expected Time Complexity: O(n+m)
Expected Auxilliary Space: O(n+m)

Note: **Companies** :
Zoho, Rockstand

Ans:

```java
import java.util.HashSet;

public class Solution {
   public static int doUnion(int a[], int n, int b[], int m) {
       HashSet<Integer> unionSet = new HashSet<>();

       for (int i = 0; i < n; i++) {
           unionSet.add(a[i]);
```

```
        }

        for (int i = 0; i < m; i++) {
            unionSet.add(b[i]);
        }

        return unionSet.size();
    }

    public static void main(String[] args) {
        int[] arr1 = {1, 2, 3, 4, 5};
        int[] arr2 = {1, 2, 3};
        int n1 = arr1.length;
        int m1 = arr2.length;
        System.out.println("Count of union elements for arr1 and arr2: " +
doUnion(arr1, n1, arr2, m1));

        int[] arr3 = {85, 25, 1, 32, 54, 6};
        int[] arr4 = {85, 2};
        int n2 = arr3.length;
        int m2 = arr4.length;
        System.out.println("Count of union elements for arr3 and arr4: " +
doUnion(arr3, n2, arr4, m2));
    }
}
```