QUESTION-01

Given an unsorted array Arr of size N of positive integers. One number 'A' from set {1, 2,....,N} is missing and one number 'B' occurs twice in array. Find these two numbers.

Example 1:

Input:

N = 2

Arr[] = {2, 2}

Output: 2 1

Explanation: Repeating number is 2 and smallest positive missing number is 1.

Example 2:

Input:

N = 3

Arr[] = {1, 3, 3}

Output: 3 2

Explanation: Repeating number is 3 and smallest positive missing number is 2.

Your Task:

You don't need to read input or print anything. Your task is to complete the function findTwoElement() which takes the array of integers arr and n as parameters and returns an array of integers of size 2 denoting the answer (The first index contains B and second index contains A.)

Expected Time Complexity: O(N)

Expected Auxiliary Space: O(1)

Constraints:

2 ≤ N ≤ 105

1 ≤ Arr[i] ≤ N

Ans:

```
public class Solution {
   public static int[] findTwoElement(int[] arr, int n) {
       int[] result = new int[2];

       int i = 0;
       while (i < n) {
           if (arr[i] != arr[arr[i] - 1]) {
               swap(arr, i, arr[i] - 1);
           } else {
               i++;
           }
       }
```

```
        for (i = 0; i < n; i++) {
            if (arr[i] != i + 1) {
                result[0] = arr[i];
                result[1] = i + 1;
                break;
            }
        }

        return result;
    }

    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 = {2, 2};
        int n1 = 2;
        int[] result1 = findTwoElement(arr1, n1);
        System.out.println("Repeating number: " + result1[0] + ", Missing
number: " + result1[1]);

        int[] arr2 = {1, 3, 3};
        int n2 = 3;
        int[] result2 = findTwoElement(arr2, n2);
        System.out.println("Repeating number: " + result2[0] + ", Missing
number: " + result2[1]);
    }
}
```

QUESTION-02

Given a sorted array arr containing n elements with possibly some duplicate, the task is to find the first and last occurrences of an element x in the given array.

Note: If the number x is not found in the array then return both the indices as -1.

Example 1:

Input:

n=9, x=5

arr[] = { 1, 3, 5, 5, 5, 5, 67, 123, 125 }

Output:

2 5

Explanation:

First occurrence of 5 is at index 2 and last occurrence of 5 is at index 5.

Example 2:

Input:

n=9, x=7

arr[] = { 1, 3, 5, 5, 5, 5, 7, 123, 125 }

Output:

6 6

Explanation:

First and last occurrence of 7 is at index 6.

Your Task:

Since, this is a function problem. You don't need to take any input, as it is already accomplished by the driver code. You just need to complete the function find() that takes array arr, integer n and integer x as parameters and returns the required answer.

Expected Time Complexity: O(logN)

Expected Auxiliary Space: O(1).

Constraints:

1 ≤ N ≤ 106

1 ≤ arr[i],x ≤ 109

Ans:

```
public class Solution {
```

```java
public static int[] findTwoElement(int[] arr, int n) {
    int[] result = new int[2];

    int i = 0;
    while (i < n) {
        if (arr[i] != arr[arr[i] - 1]) {
            swap(arr, i, arr[i] - 1);
        } else {
            i++;
        }
    }

    for (i = 0; i < n; i++) {
        if (arr[i] != i + 1) {
            result[0] = arr[i];
            result[1] = i + 1;
            break;
        }
    }

    return result;
}

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 = {2, 2};
    int n1 = 2;
    int[] result1 = findTwoElement(arr1, n1);
```

```
        System.out.println("Repeating number: " + result1[0] + ", Missing
number: " + result1[1]);

        int[] arr2 = {1, 3, 3};
        int n2 = 3;
        int[] result2 = findTwoElement(arr2, n2);
        System.out.println("Repeating number: " + result2[0] + ", Missing
number: " + result2[1]);
    }
}
```

QUESTION-3

Given a sorted array of positive integers. Your task is to rearrange the array elements alternatively i.e first element should be max value, second should be min value, third should be second max, fourth should be second min and so on.

Note: Modify the original array itself. Do it without using any extra space. You do not have to return anything.

Example 1:

Input:

n = 6

arr[] = {1,2,3,4,5,6}

Output: 6 1 5 2 4 3

Explanation: Max element = 6, min = 1,

second max = 5, second min = 2, and

so on... Modified array is : 6 1 5 2 4 3.

Example 2:

Input:

n = 11

arr[]={10,20,30,40,50,60,70,80,90,100,110}

Output:110 10 100 20 90 30 80 40 70 50 60

Explanation: Max element = 110, min = 10,

second max = 100, second min = 20, and

so on... Modified array is :

110 10 100 20 90 30 80 40 70 50 60.

Your Task:

The task is to complete the function rearrange() which rearranges elements as explained above. Printing of the modified array will be handled by driver code.

Expected Time Complexity: O(N).

Expected Auxiliary Space: O(1).

Constraints:

1 <= n <= 10^6

1 <= arr[i] <= 10^7

Ans:

```
public class Solution {
   public static void rearrange(int arr[], int n) {
       int maxIdx = n - 1;
       int minIdx = 0;
       int maxElem = arr[maxIdx] + 1;

       for (int i = 0; i < n; i++) {
           if (i % 2 == 0) {
               arr[i] += (arr[maxIdx] % maxElem) * maxElem;
               maxIdx--;
           } else { // At odd index, store minimum elements
               arr[i] += (arr[minIdx] % maxElem) * maxElem;
               minIdx++;
           }
       }

       for (int i = 0; i < n; i++) {
           arr[i] /= maxElem;
       }
   }

   public static void main(String[] args) {
       int[] arr1 = {1, 2, 3, 4, 5, 6};
       int n1 = arr1.length;
```

```
        rearrange(arr1, n1);
        printArray(arr1);

        int[] arr2 = {10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110};
        int n2 = arr2.length;
        rearrange(arr2, n2);
        printArray(arr2);
    }

    public static void printArray(int[] arr) {
        for (int num : arr) {
            System.out.print(num + " ");
        }
        System.out.println();
    }
}
```

QUESTION 4

Given two sorted arrays arr1 and arr2 of size N and M respectively and an element K. The task is to find the element that would be at the kth position of the final sorted array.

Example 1:

Input:

arr1[] = {2, 3, 6, 7, 9}

arr2[] = {1, 4, 8, 10}

k = 5

Output:

6

Explanation:

The final sorted array would be -

1, 2, 3, 4, 6, 7, 8, 9, 10

The 5th element of this array is 6.

Example 2:

Input:

arr1[] = {100, 112, 256, 349, 770}

arr2[] = {72, 86, 113, 119, 265, 445, 892}

k = 7

Output:

256

Explanation:

Final sorted array is - 72, 86, 100, 112,

113, 119, 256, 265, 349, 445, 770, 892

7th element of this array is 256.

Your Task:

You don't need to read input or print anything. Your task is to complete the function kthElement() which takes the arrays arr1[], arr2[], its size N and M respectively and an integer K as inputs and returns the element at the Kth position.

Expected Time Complexity: O(Log(N) + Log(M))

Expected Auxiliary Space: O(Log (N))

Constraints:

1 <= N, M <= 106

0 <= arr1i, arr2i < INT_MAX

1 <= K <= N+M

Ans:

```
public class Solution {
   public static int kthElement(int[] arr1, int[] arr2, int n, int m, int k) {
       if (n > m) {
           return kthElement(arr2, arr1, m, n, k);
       }

       int low = Math.max(0, k - m);
       int high = Math.min(k, n);
```

```
        while (low <= high) {
            int partitionX = (low + high) / 2;
            int partitionY = k - partitionX;

            int maxX = (partitionX == 0) ? Integer.MIN_VALUE : arr1[partitionX
- 1];
            int maxY = (partitionY == 0) ? Integer.MIN_VALUE : arr2[partitionY
- 1];

            int minX = (partitionX == n) ? Integer.MAX_VALUE :
arr1[partitionX];
            int minY = (partitionY == m) ? Integer.MAX_VALUE :
arr2[partitionY];

            if (maxX <= minY && maxY <= minX) {
                return Math.max(maxX, maxY);
            } else if (maxX > minY) {
                high = partitionX - 1;
            } else {
                low = partitionX + 1;
            }
        }

        throw new IllegalArgumentException("Invalid input");
    }

    public static void main(String[] args) {
        int[] arr1 = {2, 3, 6, 7, 9};
        int[] arr2 = {1, 4, 8, 10};
        int n = arr1.length;
        int m = arr2.length;
        int k = 5;
        System.out.println("Kth element: " + kthElement(arr1, arr2, n, m, k));
```

```
        int[] arr3 = {100, 112, 256, 349, 770};
        int[] arr4 = {72, 86, 113, 119, 265, 445, 892};
        n = arr3.length;
        m = arr4.length;
        k = 7;
        System.out.println("Kth element: " + kthElement(arr3, arr4, n, m, k));
    }
}
```