231901020 KAVIYA V

week - 02

Count the occurrence

Problem Statement:

You are given an array of integers nums. You are also given an integer original

which is the first number that needs to be searched for in nums.

You then do the following steps:

If original is found in nums, multiply it by two (i.e., set original = 2 * original).

Otherwise, stop the process.

Repeat this process with the new number as long as you keep finding the number.

Return the final value of original.

Input Format

First Line : The Array size n

Second Line : Array elements one in each line

Third Line : Original number

Output Format

First Line : the number

Sample Input

5

5 3 6 1 12

3

Sample Output

24

Explanation:

- 3 is found in nums. 3 is multiplied by 2 to obtain 6.

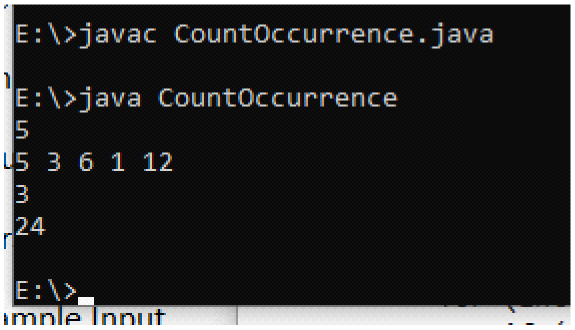
- 6 is found in nums. 6 is multiplied by 2 to obtain 12.

- 12 is found in nums. 12 is multiplied by 2 to obtain 24.

- 24 is not found in nums. Thus, 24 is returned.

coding:

```
import java.util.Scanner;

public class CountOccurrence {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        int n = scanner.nextInt();

        int[] nums = new int[n];

        for (int i = 0; i < n; i++) {

            nums[i] = scanner.nextInt();

        }

        int original = scanner.nextInt();

        while (true) {

            boolean found = false;

            for (int i = 0; i < n; i++) {

                if (nums[i] == original) {

                    original *= 2;

                    found = true;

                    break;

                }
            }
            if (!found) break;
        }
        System.out.println(original);
    }
}
```



```
E:\>javac CountOccurrence.java

E:\>java CountOccurrence
5
5 3 6 1 12
3
24

E:\>
```

2.2.Inventory Management

Problem Statement:

You are given an array prices where prices[i] is the price of a given stock on the ith day. You want to maximize your profit by choosing a single day to buy one stock and choosing a different day in the future to sell that stock.

Return the maximum profit you can achieve from this transaction. If you cannot achieve any profit, return 0.

Input Format

First line : The array size : 6

Second Line : the array element s : 7 1 5 3 6 4

Output Format

First Line : 5

Sample Input

6

7 1 5 3 6 4

Sample Output

5

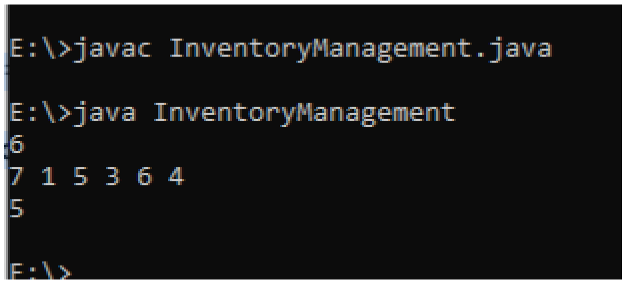
Explanation:

Buy on day 2 (price = 1) and sell on day 5 (price = 6), profit = 6-1 = 5.

Note that buying on day 2 and selling on day 1 is not allowed because you must buy before you sell.

coding:

```
import java.util.Scanner;

public class InventoryManagement {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int n = scanner.nextInt();
        int[] prices = new int[n];
        for (int i = 0; i < n; i++) {
            prices[i] = scanner.nextInt();
        }
        int minPrice = prices[0], maxProfit = 0;


        for (int i = 1; i < n; i++) {
            if (prices[i] < minPrice) {
                minPrice = prices[i];
            } else {
                maxProfit = Math.max(maxProfit, prices[i] - minPrice);
            }
        }
          System.out.println(maxProfit);
    }
}
```



```
E:\>javac InventoryManagement.java

E:\>java InventoryManagement
6
7 1 5 3 6 4
5

E:\>
```

2.3. Sort an array of 0s, 1s and 2s

Problem Statement:

Given an Array of N with the elements of 0's, 1's and 2's.

Your task is to arrange the array elements in the following order.

0's followed by 1's followed 2's

Input Format

First Line : The Array size : n

Second Line: The array elements

Output Format

Print the array elements as expected.

Sample Input 1

6

{0, 1, 2, 0, 1, 2}

Sample Output 1

{0, 0, 1, 1, 2, 2}

Explanation: {0, 0, 1, 1, 2, 2} has all 0s first, then all 1s and all 2s in last.

Input: {0, 1, 1, 0, 1, 2, 1, 2, 0, 0, 0, 1}

Output: {0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 2, 2}

Explanation: {0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 2, 2} has all 0s first, then all 1s and all 2s in last.

```
import java.util.Scanner;

public class SortArray {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int n = scanner.nextInt();
        int[] arr = new int[n];
        int count0 = 0, count1 = 0, count2 = 0;
        for (int i = 0; i < n; i++) {
            arr[i] = scanner.nextInt();
            if (arr[i] == 0) count0++;
            else if (arr[i] == 1) count1++;
            else count2++;
        }

        for (int i = 0; i < n; i++) {
            if (count0 > 0) {
                arr[i] = 0;
                count0--;
            } else if (count1 > 0) {
                arr[i] = 1;
                count1--;
            } else {
                arr[i] = 2;
            }
        }

        for (int i = 0; i < n; i++) {
            System.out.print(arr[i] + " ");
        }
    }
}
```

2.4 Find the Missing Number

Problem Statement:

Given an array arr[] of size N-1 with integers in the range of [1, N], the task is to find the missing number from the first N integers.

Note: There are no duplicates in the list.

Input Format

First Line : The array size : N

Second Line : The array elements

Output Format

Print the missing number .

SAMPLE INPUT 1

```
8
1 2 4 6 3 7 8
```

SAMPLE OUTPUT 1

```
5
```

Input: arr[] = {1, 2, 4, 6, 3, 7, 8} , N = 8

Output: 5

Explanation: Here the size of the array is 8, so the range will be [1, 8]

. The missing number between 1 to 8 is 5

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

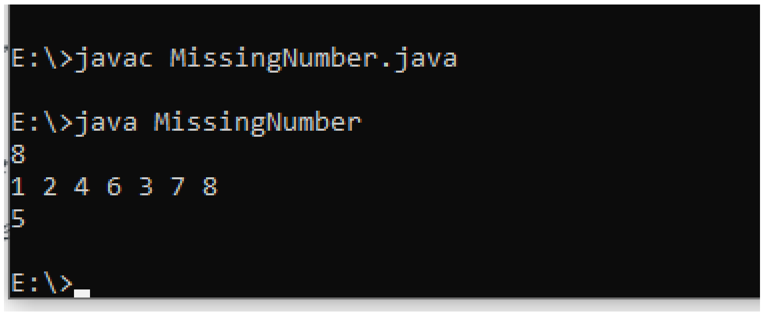
Output: 4

Explanation: Here the size of the array is 4, so the range will be [1, 5].

The missing number between 1 to 5 is 4

coding:

```java
import java.util.Scanner;


public class MissingNumber {
	public static void main(String[] args) {
		Scanner scanner = new Scanner(System.in);
		int N = scanner.nextInt();
		int[] arr = new int[N-1];
		int totalSum = N * (N + 1) / 2; // Sum of first N natural numbers
		int arraySum = 0;


		for (int i = 0; i < N-1; i++) {
			arr[i] = scanner.nextInt();
			arraySum += arr[i];
		}


		int missingNumber = totalSum - arraySum;
		System.out.println(missingNumber);
	}
}
```



```
E:\>javac MissingNumber.java

E:\>java MissingNumber
8
1 2 4 6 3 7 8
5

E:\>
```

2.5.Move all Zeroes to the End of the Array

Problem Statement:

Given an array of N elements, You task is to move the Zeroes to the end of the Array.

Input Format

First Line : Array Size : N

Second Line: Array elements separated by space

Output Format

First line: Array elements arranged as zeroes at the end.

SAMPLE INPUT

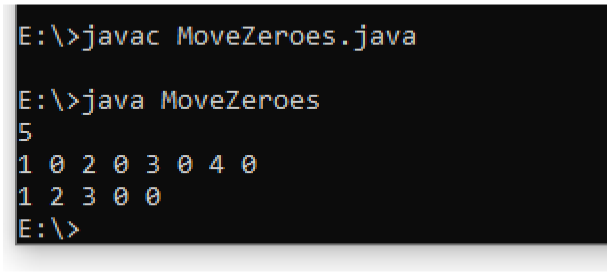
```
5
1 0 2 0 3 0 4 0
```

SAMPLE OUTPUT

```
1 2 3 4 0 0 0
```

coding:

```
import java.util.Scanner;

public class MoveZeroes {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int N = scanner.nextInt();
        int[] arr = new int[N];
        int index = 0;

        for (int i = 0; i < N; i++) {
            arr[i] = scanner.nextInt();
            if (arr[i] != 0) arr[index++] = arr[i];
        }

        while (index < N) arr[index++] = 0;

        for (int i = 0; i < N; i++) {
            System.out.print(arr[i] + " ");
        }
    }
}
```



```
E:\>javac MoveZeroes.java

E:\>java MoveZeroes
5
1 0 2 0 3 0 4 0
1 2 3 0 0
E:\>
```