Subarray with given sum

Given an unsorted array **A** of size **N** of non-negative integers, find a continuous sub-array which adds to a given number **S**.

Input:
The first line of input contains an integer **T** denoting the number of test cases. Then **T** test cases follow. Each test case consists of two lines. The first line of each test case is **N** and **S**, where N is the size of array and S is the sum. The second line of each test case contains **N** space separated integers denoting the array elements.

Output:
For each testcase, in a new line, print the **starting and ending positions**(**1** indexing) of **first such occuring subarray from the left** if sum equals to subarray, else print **-1**.

Constraints:
$1 <= T <= 100$

$1 <= N <= 10^7$

$1 <= A_i <= 10^{10}$

Example:
Input:
2
5 12
1 2 3 7 5
10 15
1 2 3 4 5 6 7 8 9 10
Output:
2 4
1 5

Explanation :
Testcase1: sum of elements from 2nd position to 4th position is 12
Testcase2: sum of elements from 1st position to 5th position is 15

Count the triplets

Given an array of distinct integers. The task is to count all the triplets such that sum of two elements equals the third element.

Input:
The first line of input contains an integer **T** denoting the number of test cases. Then T test cases follow. Each test case consists of two lines. First line of each test case contains an Integer **N** denoting size of array and the second line contains N space separated elements.

Output:
For each test case, print the count of all triplets, in new line. If no such triplets can form, print "**-1**".

Constraints:
1 <= T <= 100

3 <= N <= 10^5

1 <= A[i] <= 10^6

Example:
Input:
2
4
1 5 3 2
3
3 2 7
Output:
2
-1

Explanation:
Testcase 1: There are 2 triplets: 1 + 2 = 3 and 3 +2 = 5

Kadane's Algorithm

Given an array **arr** of **N** integers. Find the contiguous sub-array with maximum sum.

Input:
The first line of input contains an integer **T** denoting the number of test cases. The description of **T** test cases follows. The first line of each test case contains a single integer

N denoting the size of array. The second line contains **N** space-separated integers **A_1, A_2, ..., A_N** denoting the elements of the array**.**

Output:
Print the maximum sum of the contiguous sub-array in a separate line for each test case.

Constraints:
$1 \leq T \leq 110$

$1 \leq N \leq 10^6$

$-10^7 \leq A[i] <= 10^7$

Example:
Input
2
5
1 2 3 -2 5
4
-1 -2 -3 -4
Output
9
-1

Explanation:
Testcase 1: Max subarray sum is 9 of elements (1, 2, 3, -2, 5) which is a contiguous subarray.

Missing number in array

Given an array **C** of size **N-1** and given that there are numbers from **1** to **N** with one element missing, the missing number is to be found.

Input:
The first line of input contains an integer **T** denoting the number of test cases. For each test case first line contains **N**(size of array). The subsequent line contains N-1 array elements.

Output:
Print the missing number in array.

Constraints:
$1 \leq T \leq 200$

$1 \le N \le 10^7$

$1 \le C[i] \le 10^7$

Example:
Input:
2
5
1 2 3 5
10
1 2 3 4 5 6 7 8 10

Output:
4
9

Explanation:
Testcase 1: Given array : 1 2 3 5. Missing element is 4.

\

Merge Without Extra Space

Given two sorted arrays **arr1[]** and **arr2[]** in non-decreasing order with size **n** and **m**. The task is to merge the two sorted arrays into one sorted array (in non-decreasing order).

Note: Expected time complexity is O((n+m) log(n+m)). **DO NOT** use extra space. We need to modify existing arrays as following.

```
Input: arr1[] = {10};
     arr2[] = {2, 3};
Output: arr1[] = {2}
      arr2[] = {3, 10}
```

```
Input: arr1[] = {1, 5, 9, 10, 15, 20};
     arr2[] = {2, 3, 8, 13};
Output: arr1[] = {1, 2, 3, 5, 8, 9}
      arr2[] = {10, 13, 15, 20}
```

Input:
First line contains an integer **T**, denoting the number of test cases. First line of each test case contains two space separated integers **X and Y**, denoting the size of the two sorted arrays. Second line of each test case contains **X** space separated integers, denoting the first

sorted array P. Third line of each test case contains **Y** space separated integers, denoting the second array Q.

Output:
For each test case, print (**X + Y**) space separated integer representing the merged array.

Constraints:
1 <= **T** <= 100

1 <= **X, Y** <= $5*10^4$

0 <= $\mathbf{arr1_i}$, $\mathbf{arr2_i}$ <= 10^9

Example:
Input:
2
4 5
1 3 5 7
0 2 6 8 9
2 3
10 12
5 18 20

Output:
0 1 2 3 5 6 7 8 9
5 10 12 18 20

Explanation:
Testcase 1: After merging two non-decreasing arrays, we have, 0 1 2 3 5 6 7 8 9.

Rearrange Array Alternately

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: O(1) extra space is allowed. Also, try to modify the input array as required.

Input:
First line of input conatins number of test cases **T**. First line of test case contain an integer denoting the array size **N** and second line of test case contain N space separated integers denoting the array elements.

Output:
Output the modified array with alternated elements.

Constraints:
1 <=T<= 100

1 <=N<= 10^7

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

Example:
Input:
2
6
1 2 3 4 5 6
11
10 20 30 40 50 60 70 80 90 100 110

Output:
6 1 5 2 4 3
110 10 100 20 90 30 80 40 70 50 60

Explanation:
Testcase 1: Max element = 6, min = 1, second max = 5, second min = 2, and so on...
Modified array is : 6 1 5 2 4 3.

Number of pairs

Given two arrays **X** and **Y** of positive integers, find number of pairs such that $\mathbf{x^y > y^x}$ **(raised to power of)** where x is an element from X and y is an element from Y.

Input:
The first line of input contains an integer **T**, denoting the number of test cases. Then T test cases follow. Each test consists of three lines. The first line of each test case consists of two space separated M and N denoting size of arrays X and Y respectively. The second line of each test case contains M space separated integers denoting the elements of array X. The third line of each test case contains N space separated integers denoting elements of array Y.

Output:

Corresponding to each test case, print in a new line, the number of pairs such that $\mathbf{x^y > y^x}$.

Constraints:
$1 \leq T \leq 100$

$1 \leq M, N \leq 10^5$

$1 \leq X[i], Y[i] \leq 10^3$

Example:
Input
1
3 2
2 1 6
1 5

Output
3

Explanation:

Testcase 1: The pairs which follow $x^y > y^x$ are as such: $2^1 > 1^2$, $2^5 > 5^2$ and $6^1 > 1^6$

Inversion of array

Given an array of positive integers. The task is to find inversion count of array.

Inversion Count *:* For an array, inversion count indicates how far (or close) the array is from being sorted. If array is already sorted then inversion count is 0. If array is sorted in reverse order that inversion count is the maximum.
Formally, two elements a[i] and a[j] form an inversion if a[i] > a[j] and i < j.

Input:
The first line of input contains an integer T denoting the number of test cases. The first line of each test case is N, the size of array. The second line of each test case contains N elements.

Output:
Print the inversion count of array.

Constraints:
$1 \leq T \leq 100$

$1 \leq N \leq 10^7$

$1 \leq C \leq 10^{18}$

Example:
Input:
1
5
2 4 1 3 5

Output:
3

Explanation:
Testcase 1: The sequence 2, 4, 1, 3, 5 has three inversions (2, 1), (4, 1), (4, 3).

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

Given an array **A** of size **N** containing **0s, 1s, and 2s**; you need to sort the array in ascending order.

Input:
The first line contains an integer **'T'** denoting the total number of test cases. Then **T** testcases follow. Each testcases contains two lines of input. The first line denotes the size of the array **N**. The second lines contains the elements of the array A separated by spaces.

Output:
For each testcase, print the sorted array.

Constraints:
1 <= T <= 500
1 <= N <= 10^6
0 <= A_i <= 2

Example:
Input :
2
5
0 2 1 2 0
3
0 1 0

Output:
0 0 1 2 2
0 0 1

Explanation:
Testcase 1: After segragating the 0s, 1s and 2s, we have 0 0 1 2 2 which shown in the output.

Equilibrium point

Given an array **A** of **N** positive numbers. The task is to find the position where equilibrium first occurs in the array. Equilibrium position in an array is a position such that the sum of elements before it is equal to the sum of elements after it.

Input:
The first line of input contains an integer **T**, denoting the number of test cases. Then T test cases follow. First line of each test case contains an integer N denoting the size of the array. Then in the next line are N space separated values of the array A.

Output:
For each test case in a new line print the position at which the elements are at equilibrium if no equilibrium point exists print -1.

Constraints:
1 <= T <= 100

1 <= N <= 10^6

1 <= A_i <= 10^8

Example:
Input:
2
1
1
5
1 3 5 2 2

Output:
1
3

Explanation:
Testcase 1: Since its the only element hence its the only equilibrium point.
Testcase 2: For second test case equilibrium point is at position 3 as elements below it (1+3) = elements after it (2+2).