05 - List in Python

Ex. No. : 5.1 **Date:**

Register No.: 230701324 **Name:** Someash RV

Balanced Array

Given an array of numbers, find the index of the smallest array element (the pivot), for which the sums of all elements to the left and to the right are equal. The array may not be reordered.

Example
arr=[1,2,3,4,6]

- the sum of the first three elements, 1+2+3=6. The value of the last element is 6.
- Using zero based indexing, arr[3]=4 is the pivot between the two subarrays.
- The index of the pivot is 3.

Constraints

- $3 \leq n \leq 10^5$
- $1 \leq arr[i] \leq 2 \times 10^4$, where $0 \leq i < n$
- It is guaranteed that a solution always exists.

The first line contains an integer n, the size of the array arr.
Each of the next n lines contains an integer, arr[i], where $0 \leq i < n$.
Sample Case 0
Sample Input 0
4
1
2
3
3
Sample Output 0
2
Explanation 0
The sum of the first two elements, 1+2=3. The value of the last element is 3
Using zero based indexing, arr[2]=3 is the pivot between the two subarrays
The index of the pivot is 2
Sample Case 1
Sample Input 1
3
1
2
1
Sample Output 1

1
Explanation 1
The first and last elements are equal to 1
Using zero based indexing, arr[1]=2 is the pivot between the two subarrays
The index of the pivot is 1.

For example:

Input	Result
4 1 2 3 3	2
3 1 2 1	1

Program:

```
a=int(input())

l=[]

for i in range(a):

    c=int(input())

    l.append(c)

for i in range(1,a):

    d=sum(l[0:i])

    r=sum(l[i+1:])

    if(d==r):

        print(i)
```

Output:

	Input	Expected	Got	
✔	4 1 2 3 3	2	2	✔
✔	3 1 2 1	1	1	✔

Passed all tests! ✔

Ex. No. : 5.2 **Date:**

Register No.: **Name:**

Check pair with difference k

Given an array A of sorted integers and another non negative integer k, find if there exists 2 indices i and j such that A[i] - A[j] = k, i != j.

Input Format

1. First line is number of test cases T. Following T lines contain:
2. N, followed by N integers of the array
3. The non-negative integer k

Output format

Print 1 if such a pair exists and 0 if it doesn't

Input
1
3
1
3
5
4
Output:
1
Input
1
3
1
3
5
99
Output
0

For example:

Input	Result
1 3 1 3 5 4	1

Input	Result
1 3 1 3 5 99	0

Program:

```
a=int(input())
while(a!=0):
    b=int(input())
    l=[]
    f=0
    for i in range(b):
        c=int(input())
        l.append(c)
    k=int(input())
    a-=1
    for i in range(b):
        for j in range(b):
            if(l[i]-l[j]==k and i!=j):
                f=1
                break
    if(f==1):
        print(1)
    else:
        print(0)
```

Output:

	Input	Expected	Got	
✔	1 3 1 3 5 4	1	1	✔
✔	1 3 1 3 5 99	0	0	✔

Passed all tests! ✔

Ex. No. : 5.3 **Date:**

Register No.: **Name:**

Count Elements

Complete the program to count frequency of each element of an array. Frequency of a particular element will be printed once.

Sample Test Cases

Test Case 1

Input

7

23

45

23

56

45

23

40

Output

23 occurs 3 times

45 occurs 2 times

56 occurs 1 times

40 occurs 1 times

Program:

```
import collections

def CountFrequency(arr):

    return collections.Counter(arr)
```

```
if __name__ == "__main__":
    # Input size of array
    n = int(input())

    # Input elements in array
    arr = []
    for _ in range(n):
    ele = int(input())
    arr.append(ele)

    # Calculate frequency of each element
    freq = CountFrequency(arr)


    for key, value in freq.items():
    print(f"{key} occurs {value} times")
```

Output:

	Input	Expected	Got
✔	7 23 45 23 56 45 23 40	23 occurs 3 times 45 occurs 2 times 56 occurs 1 times 40 occurs 1 times	23 occurs 45 occurs 56 occurs 40 occurs

Passed all tests! ✔

Ex. No. : 5.4 **Date:**

Register No.: **Name:**

Distinct Elements in an Array

Program to print all the distinct elements in an array. Distinct elements are nothing but the unique (non-duplicate) elements present in the given array.
Input Format:
First line take an Integer input from stdin which is array length n.
Second line take n Integers which is inputs of array.
Output Format:
Print the Distinct Elements in Array in single line which is space Separated

Example Input:
5
1
2
2
3
4
Output:
1 2 3 4
Example Input:
6
1
1
2
2
3
3
Output:
1 2 3

For example:
Input Result
5
1
2
2
3
4
1 2 3 4
6
1

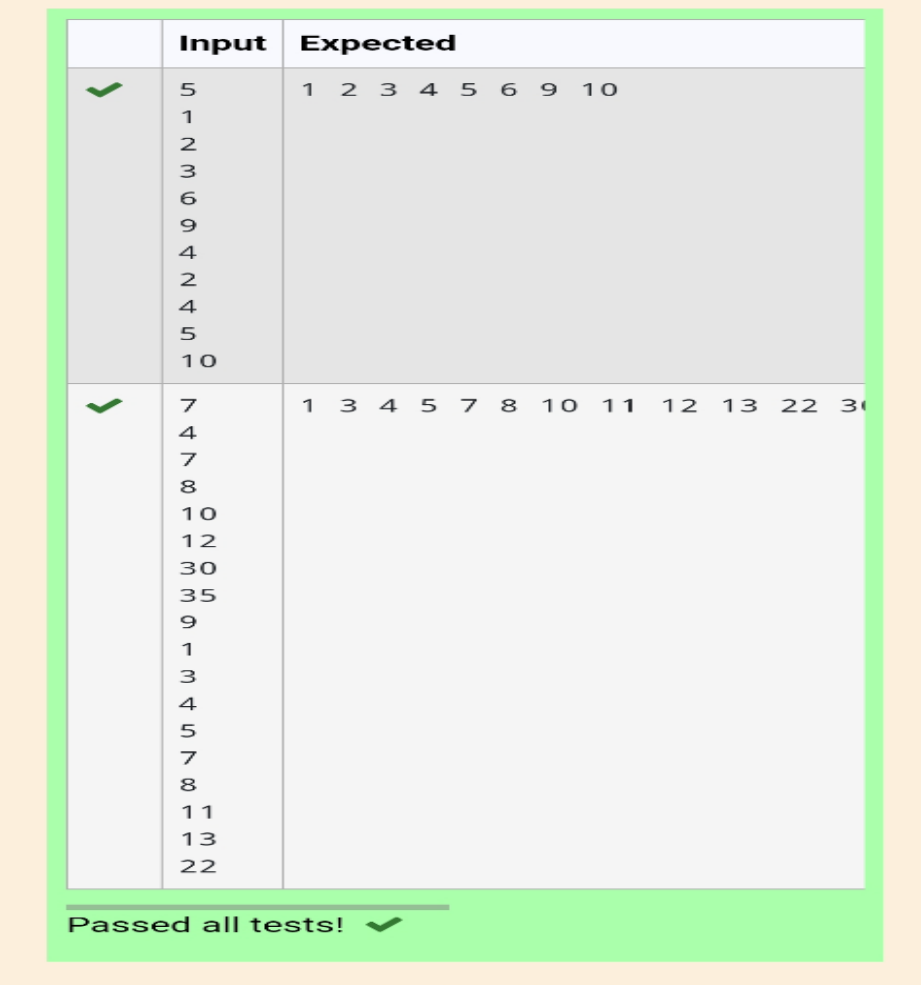
```
1
2
2
3
3
1 2 3
```

Program:

```
def merge_arrays_without_duplicates(arr1, arr2):
    # Combine the arrays and convert to a set to remove duplicates
    result_set = set(arr1 + arr2)
    # Convert the set back to a sorted list
    merged_sorted_array = sorted(result_set)
    return merged_sorted_array
# Input read and processing
def process_input():
    # Reading number of elements and the elements for the first array
    n1 = int(input())
    array1 = []
    for _ in range(n1):
        element = int(input())
        array1.append(element)
    # Reading number of elements and the elements for the second array
    n2 = int(input())
    array2 = []
    for _ in range(n2):
        element = int(input())
```

```
    array2.append(element)
# Merge the arrays without duplicates
result = merge_arrays_without_duplicates(array1, array2)
# Print the result
print(" ".join(map(str, result)))
```

Output:



	Input	Expected
✔	5 1 2 3 6 9 4 2 4 5 10	1 2 3 4 5 6 9 10
✔	7 4 7 8 10 12 30 35 9 1 3 4 5 7 8 11 13 22	1 3 4 5 7 8 10 11 12 13 22 3(

Passed all tests! ✔

Ex. No. : 5.5 **Date:**

Register No.: **Name:**

Element Insertion

Consider a program to insert an element / item in the sorted array. Complete the logic by filling up required code in editable section. Consider an array of size 10. The eleventh item is the data is to be inserted.

Sample Test Cases
Test Case 1
Input
1
3
4
5
6
7
8
9
10
11
2

Output
ITEM to be inserted:2
After insertion array is:
1
2
3
4
5
6
7
8
9
10
11

Test Case 2
Input
11
22
33
55
66
77
88
99
110
120
44

Output

ITEM to be inserted:44
After insertion array is:
11
22
33
44
55
66
77
88
99
110
120

Program:

```
def insert_sorted(list, n):

    list.append(n)

    sorted_list = sorted(list)

    print("After insertion array is:")

    for i in range(11):

        print(sorted_list[i])


sorted_list = [int(input()) for i in
range(10)]

new_element = int(input())

print("ITEM to be inserted:",
new_element, sep='')

insert_sorted(sorted_list,
new_element)
```

Output:

	Input	Expected	G
✔	1 3 4 5 6 7 8 9 10 11 2	ITEM to be inserted:2 After insertion array is: 1 2 3 4 5 6 7 8 9 10 11	IT Af 1 2 3 4 5 6 7 8 9 1(11
✔	11 22 33 55 66 77 88 99 110 120 44	ITEM to be inserted:44 After insertion array is: 11 22 33 44 55 66 77 88 99 110 120	IT Af 11 22 33 44 55 66 77 88 99 11 12

Passed all tests! ✔

Ex. No. : 5.6 **Date:**

Register No.: **Name:**

Find the Factor

Determine the factors of a number (i.e., all positive integer values that evenly divide into a number) and then return the p^{th} element of the list, sorted ascending. If there is no p^{th} element, return 0.

Constraints

$1 \leq n \leq 10^{15}$

$1 \leq p \leq 10^{9}$

The first line contains an integer n, the number to factor.

The second line contains an integer p, the 1-based index of the factor to return.

Sample Case 0
Sample Input 0
10
3
Sample Output 0
5
Explanation 0
Factoring n = 10 results in {1, 2, 5, 10}. Return the p = 3rd factor, 5, as the answer.
Sample Case 1
Sample Input 1
10
5
Sample Output 1
0
Explanation 1
Factoring n = 10 results in {1, 2, 5, 10}. There are only 4 factors and p = 5, therefore 0 is returned as the answer.
Sample Case 2
Sample Input 2
1
1
Sample Output 2
1
Explanation 2
Factoring n = 1 results in {1}. The p = 1st factor of 1 is returned as the answer.

For example:

Input	Result
10 3	5
10 5	0
1 1	1

Program:

```
import sys

import math


def find_factors(n):

        factors = []

        for i in range(1, int(math.sqrt(n)) + 1):

        if n % i == 0:

        factors.append(i)

        if i != n // i:

                factors.append(n // i)

        return sorted(factors)


def get_pth_factor(n, p):

        factors = find_factors(n)

        if p <= len(factors):

        return factors[p - 1]

        else:

        return 0
```

```
# Reading input directly from the standard input (typically for competitive programming)

input = sys.stdin.read

data = input().split()

n = int(data[0])

p = int(data[1])


# Calculate and print the p-th factor

print(get_pth_factor(n, p))
```

Output:

	Input	Expected	Got	
✔	10 3	5	5	✔
✔	10 5	0	0	✔
✔	1 1	1	1	✔

Passed all tests! ✔

Ex. No. : 5.7 **Date:**

Register No.: **Name:**

Merge List

Write a Python program to Zip two given lists of lists.

Input:
m : row size
n: column size
list1 and list 2 : Two lists
Output
Zipped List : List which combined both list1 and list2

Sample test case

Sample input
2
2
1
3
5
7
2
4
6
8

Sample Output

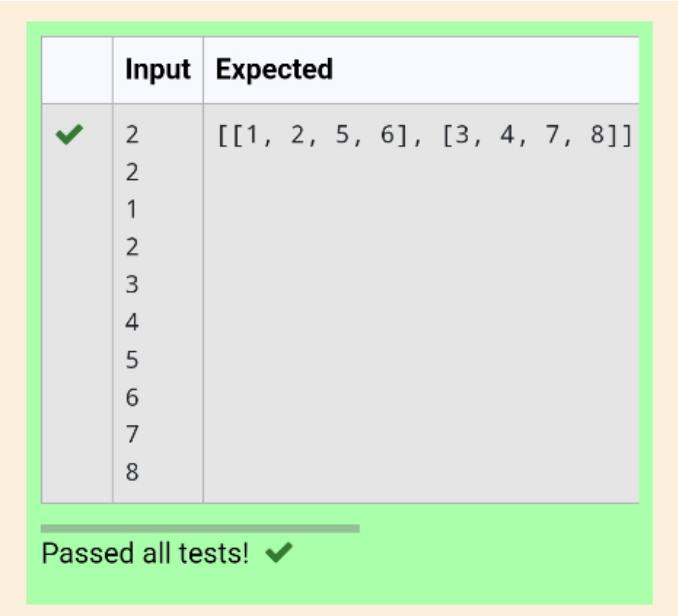
[[1, 3, 2, 4], [5, 7, 6, 8]]

Program:

```
def zip_lists(list1, list2):
    return [row1 + row2 for row1, row2 in zip(list1, list2)]


def main():
    m = int(input())
```

```
    n = int(input())


    list1 = [[int(input()) for _ in range(n)] for _ in range(m)]
    list2 = [[int(input()) for _ in range(n)] for _ in range(m)]

    zipped_list = zip_lists(list1, list2)
    print(zipped_list)

if __name__ == "__main__":
    main()
```

Output:



Ex. No. : 5.8 **Date:**

Register No.: **Name:**

Merge Two Sorted Arrays Without Duplication

Output is a merged array without duplicates.

Input Format
N1 - no of elements in array 1
Array elements for array 1
N2 - no of elements in array 2
Array elements for array2
Output Format
Display the merged array

Sample Input 1

5
1
2
3
6
9
4
2
4
5
10

Sample Output 1

1 2 3 4 5 6 9 10

Program:

```
def merge_arrays_without_duplicates(arr1, arr2):
    # Combine the arrays and convert to a set to remove duplicates
    result_set = set(arr1 + arr2)
    # Convert the set back to a sorted list
    merged_sorted_array = sorted(result_set)
    return merged_sorted_array


# Input read and processing
```

```
def process_input():
    # Reading number of elements and the elements for the first array
    n1 = int(input())
    array1 = []
    for _ in range(n1):
        element = int(input())
        array1.append(element)

    # Reading number of elements and the elements for the second array
    n2 = int(input())
    array2 = []
    for _ in range(n2):
        element = int(input())
        array2.append(element)

    # Merge the arrays without duplicates
    result = merge_arrays_without_duplicates(array1, array2)

    # Print the result
    print(" ".join(map(str, result)))
```

Output:

	Input	Expected
✔	5 1 2 3 6 9 4 2 4 5 10	1 2 3 4 5 6 9 10
✔	7 4 7 8 10 12 30 35 9 1 3 4 5 7 8 11 13 22	1 3 4 5 7 8 10 11 12 13 22 3(

Passed all tests! ✔

Ex. No. : 5.9 **Date:**

Register No.: **Name:**

Print Element Location

Write a program to print all the locations at which a particular element (taken as input) is found in a list and also print the total number of times it occurs in the list. The location starts from 1.

For example, if there are 4 elements in the array:
5
6
5
7

If the element to search is 5 then the output will be:
5 is present at location 1
5 is present at location 3
5 is present 2 times in the array.
Sample Test Cases

Test Case 1
Input

4
5
6
5
7
5

Output

5 is present at location 1.
5 is present at location 3.
5 is present 2 times in the array.

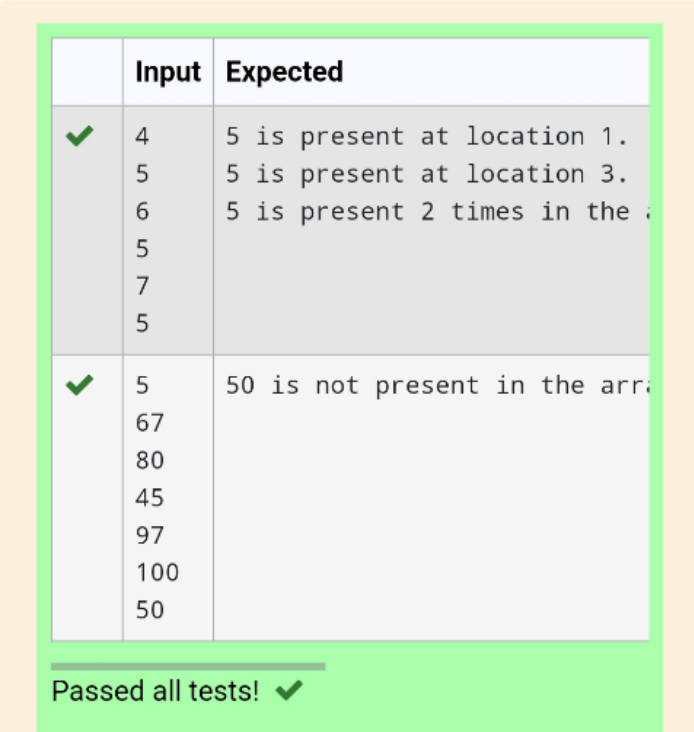
Test Case 2
Input
5
67
80
45
97
100
50

Output
50 is not present in the array.

Program:

```
def find_element_locations(lst, target):
    locations = []
    count = 0
    for i in range(len(lst)):
        if lst[i] == target:
            locations.append(i + 1)
            count += 1

    return locations, count

def main():
    n = int(input())
    lst = [int(input()) for _ in range(n)]
    target = int(input())

    locations, count = find_element_locations(lst, target)

    if count == 0:
        print(f"{target} is not present in the array.")
    else:
        for loc in locations:
            print(f"{target} is present at location {loc}.")
        print(f"{target} is present {count} times in the array.")

if __name__ == "__main__":
    main()
```

Output:



	Input	Expected
✔	4 5 6 5 7 5	5 is present at location 1. 5 is present at location 3. 5 is present 2 times in the a
✔	5 67 80 45 97 100 50	50 is not present in the arra

Passed all tests! ✔

Ex. No. : 5.10 **Date:**

Register No.: **Name:**

Strictly increasing

Write a Python program to check if a given list is strictly increasing or not. Moreover, If removing only one element from the list results in a strictly increasing list, we still consider the list true
Input:
n : Number of elements
List1: List of values
Output
Print "True" if list is strictly increasing or decreasing else print "False"

Sample Test Case

Input

7

1

2

3

0

4

5

6

Output

True

Program:

```
n= int(input())

arr = [int(input()) for i in range(n)]

l = arr.copy()
```

```
g=0
size = len(arr)
arr_asc = sorted(arr)
arr_des = sorted(arr)[::-1]
if arr==arr_asc or arr==arr_des:
    print('True')
    g=1
else:
    for i in arr:
        l.remove(i)
        arr_asc.remove(i)
        arr_des.remove(i)
        if l==arr_asc or l==arr_des:
            print('True')
            g=1
            break
        l=arr.copy()
        arr_asc = sorted(arr)
        arr_des = sorted(arr)[::-1]
if g==0:
    print('False')
```

Output:

	Input	Expected	Got	
✔	7 1 2 3 0 4 5 6	True	True	✔
✔	4 2 1 0 -1	True	True	✔

Passed all tests! ✔