Week-6

1. 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 ≤ n ≤ 105

·         1 ≤ arr[i] ≤ 2 × 104, where 0 ≤ 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 ≤ 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:

n= int(input())

arr =[]

for \_ in range(n):

arr.append(int(input()))

total\_sum = sum(arr)

left\_sum=0

for i in range(n):

right\_sum=total\_sum- left\_sum-arr[i]

if left\_sum== right\_sum:

print(i)

break

left\_sum+=arr[i]

output:

|  | **Input** | **Expected** | **Got** |  |
| --- | --- | --- | --- | --- |
|  | 4  1  2  3  3 | 2 | 2 |  |
|  | 3  1  2  1 | 1 | 1 |  |

2) 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:

n=int(input())

arr=[]

for \_ in range(n):

arr.append(int(input()))

fre={}

for num in arr:

if num in fre:

fre[num]+=1

else:

fre[num]=1

for num,fr in fre.items():

print(f"{num} occurs {fr} 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 3 times  45 occurs 2 times  56 occurs 1 times  40 occurs 1 times |  |

3) 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:

n=int(input())

my\_list =[int(input()) for \_ in range(n)]

search\_element = int(input())

locations = []

count = 0

for i in range(len(my\_list)):

if my\_list[i] == search\_element:

locations.append(i + 1)

count += 1

if count == 0:

print(f"{search\_element} is not present in the array.")

else:

for loc in locations:

print(f"{search\_element} is present at location {loc}.")

print(f"{search\_element} is present {count} times in the array.")

output:

| **Input** | **Expected** | **Got** |  |
| --- | --- | --- | --- |
|  | 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 array. | 5 is present at location 1.  5 is present at location 3.  5 is present 2 times in the array. |  |
|  | 5  67  80  45  97  100  50 | 50 is not present in the array. | 50 is not present in the array. |  |

4) 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:

n=int(input())

array=[int(input())for \_ in range(n)]

distinct\_elements = set()

for num in array:

distinct\_elements.add(num)

print(" ".join(map(str,distinct\_elements)))

output:

|  | **Input** | **Expected** | **Got** |  |
| --- | --- | --- | --- | --- |
|  | 5  1  2  2  3  4 | 1 2 3 4 | 1 2 3 4 |  |
|  | 6  1  1  2  2  3  3 | 1 2 3 | 1 2 3 |  |

5) 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.

Example

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 |
| 1  3  1  3  5  99 | 0 |

Program:

T=int(input())

for \_ in range(T):

N=int(input())

A=[int(input())for \_ in range(N)]

k=int(input())

pair\_exists= False

for i in range(N):

for j in range(i+1,N):

if A[i]-A[j]==k or A[j]-A[i]==k:

pair\_exists=True

break

if pair\_exists:

break

if pair\_exists:

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 |  |

6) 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(arr1, arr2):

merged\_array = arr1 + arr2

return list(set(merged\_array))

n1 = int(input())

arr1 = [int(input()) for \_ in range(n1)]

n2 = int(input())

arr2 = [int(input()) for \_ in range(n2)]

merged\_array = merge\_arrays(arr1, arr2)

print(\*sorted(merged\_array))

output:

InputExpectedGot

5

1

2

3

6

9

4

2

4

5

10

1 2 3 4 5 6 9 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 30 35

1 3 4 5 7 8 10 11 12 13 22 30 35

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

**Example**

n = 20

p = 3

The factors of 20 in ascending order are {1, 2, 4, 5, 10, 20}. Using 1-based indexing, if p = 3, then 4 is returned. If p > 6, 0 would be returned.

**Constraints**

1 ≤ n ≤ 1015

1 ≤ p ≤ 109

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:

n= int(input())

p= int(input())

factors = []

for i in range(1,n+1):

if n % i== 0:

factors.append(i)

factors.sort()

if p <=len(factors):

print(factors [p -1])

else:

print(0)

output:

|  | **Input** | **Expected** | **Got** |  |
| --- | --- | --- | --- | --- |
|  | 10  3 | 5 | 5 |  |
|  | 10  5 | 0 | 0 |  |
|  | 1  1 | 1 | 1 |  |

Passed all tests!

8) 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(""))

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

def is\_strictly\_increasing(lst):

count = 0

for i in range(1, len(lst)):

if lst[i] <= lst[i - 1]:

count += 1

if count > 1:

return False

# Check if removing the current or previous element helps

if i == 1 or lst[i] > lst[i - 2]:

continue

elif i < len(lst) - 1 and lst[i + 1] > lst[i - 1]:

continue

else:

return False

return True

def is\_strictly\_decreasing(lst):

reversed\_lst = lst[::-1]

return is\_strictly\_increasing(reversed\_lst)

if is\_strictly\_increasing(list1) or is\_strictly\_decreasing(list1):

print("True")

else:

print("False")

output:

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

9) 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:

m=int(input().strip())

n=int(input().strip())

list1=[]

list2=[]

for i in range(m):

row=[]

for j in range(n):

num=int(input().strip())

row.append(num)

list1.append(row)

for i in range(m):

row=[]

for j in range(n):

num=int(input().strip())

row.append(num)

list2.append(row)

zipped\_list=[]

for i in range(m):

zipped\_row=list1[i]+list2[i]

zipped\_list.append(zipped\_row)

print(zipped\_list)

output:

| **Input** | **Expected** | **Got** |  |
| --- | --- | --- | --- |
|  | 2  2  1  2  3  4  5  6  7  8 | [[1, 2, 5, 6], [3, 4, 7, 8]] | [[1, 2, 5, 6], [3, 4, 7, 8]] |  |

10) Given two lists A and B, and B is an anagram of A. B is an anagram of A means B is made by randomizing the order of the elements in A.

We want to find an *index mapping* P, from A to B. A mapping P[i] = j means the ith element in A appears in B at index j.

These lists A and B may contain duplicates. If there are multiple answers, output any of them.

For example, given

**Input**

5

12 28 46 32 50

50 12 32 46 28

**Output**

1 4 3 2 0

**Explanation**

A = [12, 28, 46, 32, 50]

B = [50, 12, 32, 46, 28]

We should return

[1, 4, 3, 2, 0]

as P[0] = 1 because the 0th element of A appears at B[1], and P[1] = 4 because the 1st element of A appears at B[4], and so on.

**Note:**

1. A, B have equal lengths in range [1, 100].
2. A[i], B[i] are integers in range [0, 10^5].

Program:

length = int(input())

A = list(map(int, input().split()))

B = list(map(int, input().split()))

P = [0] \* length

index\_map = {}

for i in range(length):

index\_map[B[i]] = i

for i in range(length):

P[i] = index\_map[A[i]]

print(" ".join(map(str, P)))

output:

| **Input** | **Expected** | **Got** |  |
| --- | --- | --- | --- |
|  | 5  12 28 46 32 50  50 12 32 46 28 | 1 4 3 2 0 | 1 4 3 2 0 |  |