Question **1**

Correct

Mark 1.00 out of 1.00

Flag question

Question text

Find the intersection of two sorted arrays.

OR in other words,

Given 2 sorted arrays, find all the elements which occur in both the arrays.

Input Format

The first line contains T, the number of test cases. Following T lines contain:

1.      Line 1 contains N1, followed by N1 integers of the first array

2.      Line 2 contains N2, followed by N2 integers of the second array

Output Format

The intersection of the arrays in a single line

Example

Input:

1

3 10 17 57

6 2 7 10 15 57 246

Output:

10 57

Input:

1

7

1

2

3

3

4

5

6

2

1

6

Output:

1 6

**For example:**

| **Input** | **Result** |
| --- | --- |
| 1  3  10  17  57  6  2  7  10  15  57  246 | 10 57 |
| 1  7  1  2  3  3  4  5  6  2  1  6 | 1 6 |

Answer:(penalty regime: 0 %)

t=int(input())

a=int(input())

l=[]

s=[]

t=[]

for i in range(a):

l.append(int(input()))

l.sort()

b=int(input())

for i in range(b):

s.append(int(input()))

s.sort()

for i in range(a):

for j in range(b):

if(l[i]==s[j]):

t.append(l[i])

print(\*t,sep=" ")

Feedback

|  | **Input** | **Expected** | **Got** |  |
| --- | --- | --- | --- | --- |
|  | 1  3  10  17  57  6  2  7  10  15  57  246 | 10 57 | 10 57 |  |
|  | 1  7  1  2  3  3  4  5  6  2  1  6 | 1 6 | 1 6 |  |

Passed all tests!

**Correct**

Marks for this submission: 1.00/1.00.

Question **2**

Correct

Mark 1.00 out of 1.00

Flag question

Question text

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

Answer:(penalty regime: 0 %)

a=int(input())

a1=[]

for i in range(a):

a1.append(int(input()))

b=int(input())

b1=[]

for i in range(b):

b1.append(int(input()))

c=list(set((a1+b1)))

c.sort()

print(\*c, sep=" ")

Feedback

|  | **Input** | **Expected** | **Got** |  |
| --- | --- | --- | --- | --- |
|  | 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 |  |

Passed all tests!

**Correct**

Marks for this submission: 1.00/1.00.

Question **3**

Incorrect

Mark 0.00 out of 1.00

Flag question

Question text

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 |

Answer:(penalty regime: 0 %)

# Function to check if there exists two indices i and j such that A[i] - A[j] = k, i != j

def find\_indices\_difference\_k(arr, k):

left = 0

right = 1

n = len(arr)

while right < n:

diff = arr[right] - arr[left]

# If difference is k and indices are different

if diff == k and left != right:

return 1

# If difference is less than k, increase right pointer

elif diff < k:

right += 1

# If difference is greater than k, increase left pointer

else:

left += 1

# Ensure left and right pointers are not equal

if left == right:

right += 1

# If no such pair found

return 0

# Input

t = int(input())

for \_ in range(t):

n = int(input()) # Length of the array

arr = list(map(int, input().split())) # Array elements

k = int(input()) # Non-negative integer k

# Check if such a pair exists

result = find\_indices\_difference\_k(arr, k)

# Output

print(result)

Feedback

|  | **Input** | **Expected** | **Got** |  |
| --- | --- | --- | --- | --- |
|  | 1  3  1  3  5  4 | 1 | 0 |  |
|  | 1  3  1  3  5  99 | 0 | 0 |  |

Some hidden test cases failed, too.  
Your code must pass all tests to earn any marks. Try again.

**Incorrect**

Marks for this submission: 0.00/1.00.

Question **4**

Correct

Mark 1.00 out of 1.00

Flag question

Question text

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 |

Answer:(penalty regime: 0 %)

a=int(input())

l=[]

for i in range(a):

l.append(int(input()))

ll=set(l)

print(\*ll,sep=" ")

Feedback

|  | **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 |  |

Passed all tests!

**Correct**

Marks for this submission: 1.00/1.00.

Question **5**

Correct

Mark 1.00 out of 1.00

Flag question

Question text

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

Answer:(penalty regime: 0 %)

32

# Function to check if the list is strictly increasing or strictly decreasing

def is\_strictly\_monotonic(lst):

n = len(lst)

increasing = True

decreasing = True

# Check if the list is strictly increasing or decreasing

for i in range(1, n):

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

increasing = False

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

decreasing = False

# If either increasing or decreasing, return True

if increasing or decreasing:

return True

# If neither increasing nor decreasing, check if removing one element results in a strictly increasing list

for i in range(1, n):

temp\_lst = lst[:i-1] + lst[i:]

if all(temp\_lst[j] < temp\_lst[j+1] for j in range(len(temp\_lst)-1)):

return True

return False

# Input

n = int(input())

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

# Output

print("True" if is\_strictly\_monotonic(lst) else "False")

Feedback

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

Passed all tests!

**Correct**

Marks for this submission: 1.00/1.00.

Question **6**

Correct

Mark 1.00 out of 1.00

Flag question

Question text

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 |

Answer:(penalty regime: 0 %)

a=int(input())

l=[]

b=0

c=0

for i in range(a):

l.append(int(input()))

for i in range(a):

b=sum(l[:i])

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

if(c==b):

print(i)

break

Feedback

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

Passed all tests!

**Correct**

Marks for this submission: 1.00/1.00.

Question **7**

Correct

Mark 1.00 out of 1.00

Flag question

Question text

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 |

Answer:(penalty regime: 0 %)

n=int(input())

p=int(input())

l=[]

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

if((n%i)==0):

l.append(i)

if(p<=len(l)):

print(l[p-1])

else:

print("0")

Feedback

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

Passed all tests!

**Correct**

Marks for this submission: 1.00/1.00.

Question **8**

Correct

Mark 1.00 out of 1.00

Flag question

Question text

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

Answer:(penalty regime: 0 %)

a=int(input())

l=[]

s=[]

for i in range(a):

l.append(int(input()))

for i in range(a):

if l[i] not in s:

count=l.count(l[i])

print(f'{l[i]} occurs {count} times')

s.append(l[i])

Feedback

|  | **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 |  |

Passed all tests!

**Correct**

Marks for this submission: 1.00/1.00.

Question **9**

Correct

Mark 1.00 out of 1.00

Flag question

Question text

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

Answer:(penalty regime: 0 %)

a=[]

for i in range(0,11):

a.append(int(input()))

print("ITEM to be inserted:",a[10],sep="")

a.sort()

print("After insertion array is:")

print(\*a,sep="\n")

Feedback

|  | **Input** | **Expected** | **Got** |  |
| --- | --- | --- | --- | --- |
|  | 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 | ITEM to be inserted:2  After insertion array is:  1  2  3  4  5  6  7  8  9  10  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 | ITEM to be inserted:44  After insertion array is:  11  22  33  44  55  66  77  88  99  110  120 |  |

Passed all tests!

**Correct**

Marks for this submission: 1.00/1.00.

Question **10**

Correct

Mark 1.00 out of 1.00

Flag question

Question text

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]]

Answer:(penalty regime: 0 %)

# Function to zip two lists of lists

def zip\_lists(list1, list2):

zipped\_list = []

for l1, l2 in zip(list1, list2):

zipped\_list.append(l1 + l2)

return zipped\_list

# Input

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)]

# Zipping lists

zipped\_list = zip\_lists(list1, list2)

# Output

print( zipped\_list)

Feedback

|  | **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]] |  |

Passed all tests!

**Correct**

Marks for this submission: 1.00/1.00.