**WEEK 10**

**Searching techniques : Linear and Binary**

Top of Form

An list contains N numbers and you want to determine whether two of the numbers sum to a given number K. For example, if the input is 8, 4, 1, 6 and K is 10, the answer is yes (4 and 6). A number may be used twice.

**Input Format**

The first line contains a single integer n , the length of list

The second line contains n space-separated integers, list[i].

The third line contains integer k.

**Output Format**

Print Yes or No.

**Sample Input**

7

0 1 2 4 6 5 3

1

**Sample Output**

Yes

**For example:**

| **Input** | **Result** |
| --- | --- |
| 5  8 9 12 15 3  11 | Yes |
| 6  2 9 21 32 43 43 1  4 | No |

Answer:(penalty regime: 0 %)

def twoSum(nums, target):

seen = set()

for num in nums:

complement = target - num

if complement in seen:

return "Yes"

seen.add(num)

return "No"

n = int(input())

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

target = int(input())

print(twoSum(nums, target))

|  | **Input** | **Expected** | **Got** |  |
| --- | --- | --- | --- | --- |
|  | 5  8 9 12 15 3  11 | Yes | Yes |  |
|  | 6  2 9 21 32 43 43 1  4 | No | No |  |
|  | 6  13 42 31 4 8 9  17 | Yes | Yes |  |

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

To find the frequency of numbers in a list and display in sorted order.

**Constraints:**

1<=n, arr[i]<=100

**Input:**

1 68 79 4 90 68 1 4 5

**output:**

 1 2

 4 2

 5 1

 68 2

 79 1

90 1

**For example:**

| **Input** | **Result** |
| --- | --- |
| 4 3 5 3 4 5 | 3 2  4 2  5 2 |

Answer:(penalty regime: 0 %)

def frequency\_sorted(nums):

freq = {}

for num in nums:

freq[num] = freq.get(num, 0) + 1

sorted\_freq = sorted(freq.items())

for key, value in sorted\_freq:

print(key, value)

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

frequency\_sorted(nums)

|  | **Input** | **Expected** | **Got** |  |
| --- | --- | --- | --- | --- |
|  | 4 3 5 3 4 5 | 3 2  4 2  5 2 | 3 2  4 2  5 2 |  |
|  | 12 4 4 4 2 3 5 | 2 1  3 1  4 3  5 1  12 1 | 2 1  3 1  4 3  5 1  12 1 |  |
|  | 5 4 5 4 6 5 7 3 | 3 1  4 2  5 3  6 1  7 1 | 3 1  4 2  5 3  6 1  7 1 |  |

Passed all tests!

**Correct**

Marks for this submission: 1.00/1.00.

Question **3**

Correct

Mark 1.00 out of 1.00

Flag question

Question text

Given an list, find peak element in it. A peak element is an element that is greater than its neighbors.

An element a[i] is a peak element if

A[i-1] <= A[i] >=a[i+1] for middle elements. [0<i<n-1]

A[i-1] <= A[i] for last element [i=n-1]

A[i]>=A[i+1] for first element [i=0]

**Input Format**

The first line contains a single integer n , the length of A .  
The second line contains n space-separated integers,A[i].

**Output Format**

**Print** peak numbers separated by space.

**Sample Input**

5

8 9 10 2 6

**Sample Output**

10 6

**For example:**

| **Input** | **Result** |
| --- | --- |
| 4  12 3 6 8 | 12 8 |

Answer:(penalty regime: 0 %)

def find\_peak(nums):

peaks = []

n = len(nums)

if n == 1:

return nums[0]

for i in range(n):

if i == 0 and nums[i] >= nums[i + 1]:

peaks.append(nums[i])

elif i == n - 1 and nums[i] >= nums[i - 1]:

peaks.append(nums[i])

elif nums[i] >= nums[i - 1] and nums[i] >= nums[i + 1]:

peaks.append(nums[i])

return peaks

n = int(input())

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

print(\*find\_peak(nums))

|  | **Input** | **Expected** | **Got** |  |
| --- | --- | --- | --- | --- |
|  | 7  15 7 10 8 9 4 6 | 15 10 9 6 | 15 10 9 6 |  |
|  | 4  12 3 6 8 | 12 8 | 12 8 |  |

Passed all tests!

**Correct**

Marks for this submission: 1.00/1.00.

Question **4**

Correct

Mark 1.00 out of 1.00

Flag question

Question text

Write a Python program for binary search.

**For example:**

| **Input** | **Result** |
| --- | --- |
| 1,2,3,5,8  6 | False |
| 3,5,9,45,42  42 | True |

Answer:(penalty regime: 0 %)

def binary\_search(arr, target):

left, right = 0, len(arr) - 1

while left <= right:

mid = left + (right - left) // 2

if arr[mid] == target:

return True

elif arr[mid] < target:

left = mid + 1

else:

right = mid - 1

return False

input\_list = sorted([int(x) for x in input().split(",")])

target\_value = int(input())

result = binary\_search(input\_list, target\_value)

print(result)

|  | **Input** | **Expected** | **Got** |  |
| --- | --- | --- | --- | --- |
|  | 1,2,3,5,8  6 | False | False |  |
|  | 3,5,9,45,42  42 | True | True |  |
|  | 52,45,89,43,11  11 | True | True |  |

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 sort a list of elements using the merge sort algorithm.

**For example:**

| **Input** | **Result** |
| --- | --- |
| 5  6 5 4 3 8 | 3 4 5 6 8 |

Answer:(penalty regime: 0 %)

def merge\_sort(arr):

if len(arr) > 1:

mid = len(arr) // 2

left\_half = arr[:mid]

right\_half = arr[mid:]

merge\_sort(left\_half)

merge\_sort(right\_half)

i = j = k = 0

# Merge the two sorted halves

while i < len(left\_half) and j < len(right\_half):

if left\_half[i] < right\_half[j]:

arr[k] = left\_half[i]

i += 1

else:

arr[k] = right\_half[j]

j += 1

k += 1

# Copy remaining elements of left\_half

while i < len(left\_half):

arr[k] = left\_half[i]

i += 1

k += 1

# Copy remaining elements of right\_half

while j < len(right\_half):

arr[k] = right\_half[j]

j += 1

k += 1

def print\_list(arr):

for num in arr:

print(num, end=' ')

print()

n = int(input())

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

merge\_sort(arr)

print\_list(arr)

|  | **Input** | **Expected** | **Got** |  |
| --- | --- | --- | --- | --- |
|  | 5  6 5 4 3 8 | 3 4 5 6 8 | 3 4 5 6 8 |  |
|  | 9  14 46 43 27 57 41 45 21 70 | 14 21 27 41 43 45 46 57 70 | 14 21 27 41 43 45 46 57 70 |  |
|  | 4  86 43 23 49 | 23 43 49 86 | 23 43 49 86 |  |

Passed all tests!

**Correct**

Marks for this submission: 1.00/1.00.