**10 - Searching & Sorting**

**Ex. No. : 10.1 Date: 29/05/2024**

**Register No.: 231401037 Name: ILANKO M**

**Bubble sort**

Bubble Sort is the simplest sorting algorithm that works by repeatedly swapping the adjacent elements if they are in wrong order. You read an list of numbers. You need to arrange the elements in ascending order and print the result. The sorting should be done using bubble sort.

**Input Format:**The first line reads the number of elements in the array. The second line reads the array elements one by one.

**Output Format:** The output should be a sorted list.

**For example:**

| **Input** | **Result** |
| --- | --- |
| 6  3 4 8 7 1 2 | 1 2 3 4 7 8 |
| 5  4 5 2 3 1 | 1 2 3 4 5 |

**Answer:**

def bubble\_sort(arr):

n = len(arr)

for i in range(n):

for j in range(0, n - i - 1):

if arr[j] > arr[j + 1]:

arr[j], arr[j + 1] = arr[j + 1], arr[j]

return arr

n = int(input())

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

sorted\_arr = bubble\_sort(arr)

print(\*sorted\_arr)

| **Input** | **Expected** | **Got** |  |
| --- | --- | --- | --- |
|  | 6  3 4 8 7 1 2 | 1 2 3 4 7 8 | 1 2 3 4 7 8 |  |
|  | 6  9 18 1 3 4 6 | 1 3 4 6 9 18 | 1 3 4 6 9 18 |  |
|  | 5  4 5 2 3 1 | 1 2 3 4 5 | 1 2 3 4 5 |  |

**Ex. No. : 10.2 Date: 29/05/2024**

**Register No.: 231401037 Name: ILANKO M**

**Sum of a given number k**

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:**

def has\_pair\_with\_sum(arr, k):

num\_set = set()

for num in arr:

complement = k - num

if complement in num\_set:

return "Yes"

num\_set.add(num)

return "No"

n = int(input())

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

k = int(input())

result = has\_pair\_with\_sum(arr, k)

print(result)

| **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 |  |

.

**Ex. No. : 10.3 Date: 29/05/2024**

**Register No.: 231401037 Name: ILANKO M**

**Peak Element**

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:**

def find\_peaks(nums): peaks = []

for i in range(len(nums)): if i == 0:

if nums[i] >= nums[i+1]: peaks.append(nums[i])

elif i == len(nums) - 1:

if nums[i] >= nums[i-1]: peaks.append(nums[i])

else:

if 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())) peaks = find\_peaks(nums)

print(' '.join(map(str, peaks)))

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **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 |  |

**Ex. No. : 10.4 Date: 29/05/2024**

**Register No.: 231401037 Name: ILANKO M**

**Binary Search**

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:**

def binary\_search(arr, target): left, right = 0, len(arr) - 1 while left <= right:

mid = (left + right) // 2 if arr[mid] == target:

return True

elif arr[mid] < target: left = mid + 1

else:

right = mid - 1 return False

arr\_input = input() target\_input = input()

.

arr = list(map(int, arr\_input.split(','))) target = int(target\_input)

.

arr.sort()

result = binary\_search(arr, target) 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 |  |

.

**Ex. No. : 10.5 Date: 29/05/2024**

**Register No.: 231401037 Name: ILANKO M**

**Frequency of Elements**

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:**

def frequencySorted(arr): freq\_dict = {}

for num in arr:

if num in freq\_dict: freq\_dict[num] += 1

else:

freq\_dict[num] = 1

sorted\_freq = sorted(freq\_dict.items()) for key, value in sorted\_freq:

print(key, value)

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **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 |  |