

Started on	Monday, 3 June 2024, 6:16 PM
State	Finished
Completed on	Monday, 3 June 2024, 6:27 PM
Time taken	11 mins 12 secs
Marks	5.00/5.00
Grade	100.00 out of 100.00

Question **1**

Correct

Mark 1.00 out of 1.00

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 %)

```
1 def binary_search(arr,x):
2     low = 0
3     high = len(arr)-1
4     while low <= high:
5         mid = (low+high)//2
6         if arr[mid] < x:
7             low = mid+1
8         elif arr[mid]>x:
9             high = mid-1
10        else:
11            return True
12        return False
13 e = list(map(int, input().split(",")))
14 ef = int(input())
15 e.sort()
16 f = binary_search(e,ef)
17 print(f)
```

	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 2

Correct

Mark 1.00 out of 1.00

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 %)

```
1 arr = list(map(int, input().split()))
2
3 freq = {}
4 for num in arr:
5     freq[num] = freq.get(num, 0) + 1
6
7
8 for key in sorted(freq.keys()):
9     print(key, freq[key])
10
```

	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	✓

	Input	Expected	Got	
✓	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

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 %)

```
1 def pair(arr, k):
2     seen = set()
3     for num in arr:
4         complement = k-num
5         if complement in seen:
6             return True
7         seen.add(num)
8     return False
9 n = int(input())
10 arr = list(map(int, input().split()))
11 k = int(input())
12 if pair(arr, k):
13     print("Yes")
14 else:
15     print("No")
```

	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 4

Correct

Mark 1.00 out of 1.00

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

$A[i-1] \leq A[i] \geq A[i+1]$ for middle elements. $[0 < i < n-1]$

$A[i-1] \leq A[i]$ for last element $[i=n-1]$

$A[i] \geq A[i+1]$ 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 %)

```

1 def find_peak_elements(arr):
2     peaks = []
3     n = len(arr)
4     if arr[0] >= arr[1]:
5         peaks.append(arr[0])
6     for i in range(1, n - 1):
7         if arr[i - 1] <= arr[i] >= arr[i + 1]:
8             peaks.append(arr[i])
9     if arr[-1] >= arr[-2]:
10        peaks.append(arr[-1])
11
12    return peaks
13 n = int(input(""))
14 arr = list(map(int, input("").split()))
15
16 peaks = find_peak_elements(arr)
17 print(" ".join(map(str, peaks)))
18

```

	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 5

Correct

Mark 1.00 out of 1.00

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: (penalty regime: 0 %)

```

1 def bubble_sort(arr):
2     n = len(arr)
3     for i in range(n):
4         for j in range(0, n-i-1):
5             if arr[j]>arr[j+1]:
6                 arr[j],arr[j+1]=arr[j+1],arr[j]
7 n = int(input())
8 arr = list(map(int,input().split()))
9 bubble_sort(arr)
10 print(' '.join(map(str,arr)))
11

```

	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	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

◀ [Week10_MCQ](#)

Jump to...

