Dashboard / My courses / PSPP/PUP / Searching techniques: Linear and Binary / Week10_Coding

Started on	Wednesday, 22 May 2024, 7:19 PM
State	Finished
Completed on	Wednesday, 22 May 2024, 7:42 PM
Time taken	22 mins 48 secs
Marks	5.00/5.00
Grade	**100.00** out of 100.00

Question **1**

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 %)

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

for num in sorted(set(arr)):
    print(num, arr.count(num))

```

	Input	Expected	Got	
✓	4 3 5 3 4 5	3 2 4 2 5 2	3 2 4 2 5 2	✓

	Input	Expected	Got	
✓	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 **2**

Correct

Mark 1.00 out of 1.00

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 %)

```python
def merge_sort(arr):
    if len(arr) <= 1:
        return arr

    mid = len(arr) // 2
    left_half = merge_sort(arr[:mid])
    right_half = merge_sort(arr[mid:])

    sorted_arr = []
    while left_half and right_half:
        if left_half[0] < right_half[0]:
            sorted_arr.append(left_half.pop(0))
        else:
            sorted_arr.append(right_half.pop(0))

    sorted_arr.extend(left_half)
    sorted_arr.extend(right_half)

    return sorted_arr

# Input
n = int(input())
arr = list(map(int, input().split()))

# Sorting using merge sort
arr = merge_sort(arr)

# Output
print(*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.

Question **3**

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] <= 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_elements(arr):
    return [arr[i] for i in range(len(arr)) if (i == 0 or arr[i] >= arr[i - 1]) and (i == len(a


# Input
n = int(input())
arr = list(map(int, input().split()))

# Output
print(*find_peak_elements(arr))


```

	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

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 %)

```python
def has_pair_with_sum(arr, k):
    seen = set()
    for num in arr:
        complement = k - num
        if complement in seen:
            return "Yes"
        seen.add(num)
    return "No"

# Input
n = int(input())
arr = list(map(int, input().split()))
k = int(input())

# Output
print(has_pair_with_sum(arr, k))

```

	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 **5**

Correct

Mark 1.00 out of 1.00

Given an listof integers, sort the array in ascending order using the *Bubble Sort* algorithm above. Once sorted, print the following three lines:

1. List is sorted in numSwaps swaps., where numSwaps is the number of swaps that took place.
2. First Element: firstElement, the *first* element in the sorted list.
3. Last Element: lastElement, the *last* element in the sorted list.

For example, given a worst-case but small array to sort: a=[6,4,1]. It took 3 swaps to sort the array. Output would be

```
Array is sorted in 3 swaps.
```

```
First Element: 1
```

```
Last Element: 6
```

Input Format

The first line contains an integer,n , the size of the list a .
The second line contains n, space-separated integers a[i].

Constraints

· 2<=n<=600

· 1<=a[i]<=2x10^6.

Output Format

You must print the following three lines of output:

1. List is sorted in numSwaps swaps., where numSwaps is the number of swaps that took place.
2. First Element: firstElement, the *first* element in the sorted list.
3. Last Element: lastElement, the *last* element in the sorted list.

Sample Input 0

3

1 2 3

Sample Output 0

List is sorted in 0 swaps.

First Element: 1

Last Element: 3

For example:

Input	Result
3 3 2 1	List is sorted in 3 swaps. First Element: 1 Last Element: 3
5 1 9 2 8 4	List is sorted in 4 swaps. First Element: 1 Last Element: 9

Answer: (penalty regime: 0 %)

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

```

```
swaps = 0
for i in range(n):
    for j in range(n - 1):
        if arr[j] > arr[j + 1]:
            arr[j], arr[j + 1] = arr[j + 1], arr[j]
            swaps += 1

print(f"List is sorted in {swaps} swaps.")
print(f"First Element: {arr[0]}")
print(f"Last Element: {arr[-1]}")
```

	Input	Expected	Got	
✓	3 3 2 1	List is sorted in 3 swaps. First Element: 1 Last Element: 3	List is sorted in 3 swaps. First Element: 1 Last Element: 3	✓
✓	5 1 9 2 8 4	List is sorted in 4 swaps. First Element: 1 Last Element: 9	List is sorted in 4 swaps. First Element: 1 Last Element: 9	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

◄ Week10_MCQ

Jump to...

Sorting ►