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

Started on	Saturday, 25 May 2024, 11:13 AM
State	Finished
Completed on	Sunday, 26 May 2024, 1:42 AM
Time taken	14 hours 29 mins
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 %)

```
n = input()
k = (input())
if k in n:
    print(True)
else:
    print(False)
```

	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

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 %)

```python
def bubble_sort(arr):
    n = len(arr)
    num_swaps = 0

    for i in range(n):

        swapped = False

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

                arr[j], arr[j + 1] = arr[j + 1], arr[j]
                num_swaps += 1
                swapped = True

        if not swapped:
            break

    return arr, num_swaps

n = int(input())
a = list(map(int, input().split()))

sorted_list, num_swaps = bubble_sort(a)

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

Question **3**

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 %)

```python
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]
n = int(input())
arr = list(map(int, input().split()))
bubble_sort(arr)
print(*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	✓

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_sum_to_k(arr,k):
    seen = set()
    for num in arr:
        complement = k - num
        if complement in seen:
            return "Yes"
        seen.add(num)
    return "No"
n = int(input())
arr = list(map(int, input().split()))
k = int(input())
print(has_sum_to_k(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

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

        merge_sort(left_half)
        merge_sort(right_half)

        i = j = k = 0
        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
        while i < len(left_half):
            arr[k] = left_half[i]
            i += 1
            k += 1

        while j < len(right_half):
            arr[k] = right_half[j]
            j += 1
            k += 1
n = int(input())
arr = list(map(int, input().split()))
merge_sort(arr)
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.

◄ Week10_MCQ

Jump to...

Sorting ►