

[Dashboard](#) / [My courses](#) / [PSPP/PUP](#) / [Searching techniques: Linear and Binary](#) / [Week10 Coding](#)

<b>Started on</b>	Friday, 7 June 2024, 10:14 PM
<b>State</b>	Finished
<b>Completed on</b>	Monday, 10 June 2024, 11:59 PM
<b>Time taken</b>	3 days 1 hour
<b>Marks</b>	4.00/5.00
<b>Grade</b>	<b>80.00</b> 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     arr.sort()
3     left,right=0,len(arr)-1
4     while left <=right:
5         mid=(left+right)//2
6         if arr[mid]==x:
7             return True
8         elif arr[mid]<x:
9             left=mid+1
10        else:
11            right=mid-1
12
13        return False
14
15 numbers=list(map(int,input().split(',')))
16 target=int(input())
17 result=binary_search(numbers,target)
18 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 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

- $2 \leq n \leq 600$
- $1 \leq a[i] \leq 2 \times 10^6$ .

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 %)

1

def bubble\_sort(arr):

2

num\_swaps=0

3

n=len(arr)

4

for i in range(n):

```

4  ▼   for i in range (n):
5       swapped= False
6  ▼   for j in range (0,n-i-1):
7  ▼       if arr[j]>arr[j+1]:
8           arr[j], arr[j+1]=arr[j+1],arr[j]
9           num_swaps += 1
10          swapped= True
11 ▼       if not swapped:
12           break
13       return num_swaps
14 n=int(input())
15 arr=list(map(int,input().split()))
16 num_swaps=bubble_sort(arr)
17 print("List is sorted in", num_swaps,"swaps.")
18 print("First Element:",arr[0])
19 print("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.

## 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 %)

```

1 a=int(input())
2 b=input().split()
3 x=list(b)
4 y=sorted(map(int,x))
5 for i in y:
6     print(i,end=" ")

```

	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

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 %)

```

1 | x=int(input())
2 | y=input().split()
3 | a=list(y)
4 | a=sorted(a)
5 | for i in a:
6 |     print(i,end=" ")

```

	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 5

Not answered

Mark 0.00 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 ||
```

◀ [Week10\\_MCQ](#)

Jump to...

Sorting ▶

