Started on	Friday, 17 May 2024, 12:33 PM
State	Finished
Completed on	Friday, 17 May 2024, 1:07 PM
Time taken	33 mins 32 secs
Marks	5.00/5.00
Grade	**50.00** out of 50.00 (**100**%)
Name	AVULA SNEYA DRITI 2022-CSD-A

Question **1**

Correct

Mark 1.00 out of 1.00

Take a complete sentence as an input and remove duplicate word in it and print (sorted order), then count all the words which have a length greater than 3 and print.

Input

we are good are we good

Output

are good we

Count = 1

For example:

Input	**Result**

`welcome to rec rec cse ece`	`cse ece rec to welcome`
	`Count = 1`

Answer: (penalty regime: 0 %)

```python
def process_sentence():
    sentence = input("")
    words = sentence.split()
    unique_words = set(words)
    sorted_unique_words = sorted(unique_words)
    result_sentence = ' '.join(sorted_unique_words)
    count = sum(1 for word in unique_words if len(word) > 3)
    print(result_sentence)
    print("Count =", count)
process_sentence()

```

	Input	**Expected**	**Got**	

✓	`we are good are we good`	`are good we`
		`Count = 1`
			`are good we`
			`Count = 1`
				✓

✓	`welcome to rec rec cse ece`	`cse ece rec to welcome`
		`Count = 1`
			`cse ece rec to welcome`
			`Count = 1`
				✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

Question **2**

Correct

Mark 1.00 out of 1.00

You are given an array of N integers, A1, A2, . . . , AN and an integer K. Return the of count of distinct numbers in all windows of size K.

Input :

1 2 1 3 4 3

3

Output :

2

3

3

2

Explanation

All windows of size K are

[1, 2, 1]

[2, 1, 3]

[1, 3, 4]

[3, 4, 3]

Answer: (penalty regime: 0 %)

```
from collections import defaultdict

def count_distinct_in_windows(arr, K):
    n = len(arr)
    if n < K:
        return []
    freq_map = defaultdict(int)
    result = []
    distinct_count = 0
    for i in range(K):
        if freq_map[arr[i]] == 0:
            distinct_count += 1
        freq_map[arr[i]] += 1

    result.append(distinct_count)
    for i in range(K, n):
        if freq_map[arr[i - K]] == 1:
            distinct_count -= 1
        freq_map[arr[i - K]] -= 1
        if freq_map[arr[i]] == 0:
            distinct_count += 1
        freq_map[arr[i]] += 1
```

	Input	Expected	Got	
✓	1 2 1 3 4 3 3	2 3 3 2	2 3 3 2	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

Question **3**

Correct

Mark 1.00 out of 1.00

Check if a set is a subset of another set.

Example:

Sample Input1:

mango apple

mango orange

mango

output1:

yes

set3 is subset of set1 and set2

```
input2:
```

```
mango orange
```

```
banana orange
```

```
grapes
```

```
output2:
```

no

Answer: (penalty regime: 0 %)

```
a=set(input())
b=set(input())
c=set(input())
if c.issubset(a):
    print("yes\nset3 is subset of set1 and set2")
else:
    print("No")
```

	Test	Input	Expected	Got	
✓	1	mango apple mango orange mango	yes set3 is subset of set1 and set2	yes set3 is subset of set1 and set2	✓
✓	2	mango orange banana orange grapes	No	No	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

Question **4**

Correct

Mark 1.00 out of 1.00

Two strings, *a* and *b*, are called anagrams if they contain all the same characters in the same frequencies. For example, the anagrams of CAT are CAT, ACT, TAC, TCA, ATC, and CTA.

Complete the function in the editor. If *a* and *b* are case-insensitive anagrams, print "Anagrams"; otherwise, print "Not Anagrams" instead.

Input Format

The first line contains a string denoting *a*.
The second line contains a string denoting *b*.

Constraints

- *1 ≤ length(a), length(b) ≤ 50*
- Strings *a* and *b* consist of English alphabetic characters.
- The comparison should NOT be case sensitive.

Output Format

Print "Anagrams" if *a* and *b* are case-insensitive anagrams of each other; otherwise, print "Not Anagrams" instead.

Sample Input 0

anagram

margana

Sample Output 0

Anagrams

Explanation 0

Character	**Frequency:** anagram	**Frequency:** margana
A or a	3	3
G or g	1	1
N or n	1	1
M or m	1	1
R or r	1	1

The two strings contain all the same letters in the same frequencies, so we print "Anagrams".

Answer: (penalty regime: 0 %)

```
a = (input()).lower()
b = (input()).lower()
if a ==b:
    print("Anagrams")
else:
    print("Not Anagrams")



```

	Input	Expected	Got	
✓	madam maDaM	Anagrams	Anagrams	✓
✓	DAD DAD	Anagrams	Anagrams	✓
✓	MAN MAM	Not Anagrams	Not Anagrams	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

Question **5**

Correct

Mark 1.00 out of 1.00

Given two lists, print all the common element of two lists.

Note: Sort the list before printing.

Examples:

```
Input :
1 2 3 4 5
5 6 7 8 9
Output :
5

Input :
1 2 3 4 5
6 7 8 9
Output :
No common elements

Input :
1 2 3 4 5 6
5 6 7 8 9
Output :
5 6
```

Answer: (penalty regime: 0 %)

```python
def find_common_elements(list1, list2):
    set1 = set(list1)
    set2 = set(list2)
    common_elements = set1.intersection(set2)

    if common_elements:
        sorted_common_elements = sorted(common_elements)
        print(' '.join(map(str, sorted_common_elements)))
    else:
        print("No common elements")

def main():
    list1 = list(map(int, input("").split()))
    list2 = list(map(int, input("").split()))
    find_common_elements(list1, list2)
main()

```

	Input	Expected	Got	
✓	1 2 3 4 5 5 6 7 8 9	5	5	✓
✓	1 2 3 4 5 6 7 8 9	No common elements	No common elements	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

◄ Week-09_MCQ

Jump to...

WEEK-09-Extra ►