Started on	Sunday, 5 May 2024, 9:10 PM
State	Finished
Completed on	Sunday, 5 May 2024, 9:24 PM
Time taken	13 mins 35 secs
Marks	5.00/5.00
Grade	**50.00** out of 50.00 (**100**%)
Name	THEJESH N S 2022-CSD-A

Question **1**

Correct

Mark 1.00 out of 1.00

write a program to identify the common item present in three different set but not on the other set and display the items in the sorted order.

input:

10 50 40 60 30

40 30 70 60 30

20 50 10 75 80

output:

20 70 75 80

Answer: (penalty regime: 0 %)

```
set1 = set(map(int, input().strip('{}').split(',')))
set2 = set(map(int, input().strip('{}').split(',')))
set3 = set(map(int, input().strip('{}').split(',')))


common_items = sorted((set1 ^ set2 ^ set3) - (set1 & set2 & set3))


output = "{" + ",".join(map(str, common_items)) + "}"
print(output)

```

	Test	Input	Expected	Got	
✔	1	{10,50,40,60,30} {40,30,70,60,65} {20,50,10,75,80}	{20,65,70,75,80}	{20,65,70,75,80}	✔
✔	2	{10,15,20,40,50} {30,20,40,10,25} {40,50,10,45,55}	{15,25,30,45,55}	{15,25,30,45,55}	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

Question **2**

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 %)

```
def is_anagram(a, b):

    a = a.lower().replace(" ", "")
    b = b.lower().replace(" ", "")


    if sorted(a) == sorted(b):
        return "Anagrams"
    else:
        return "Not Anagrams"


a = input().strip()
b = input().strip()
print(is_anagram(a, b))

```

	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 **3**

Correct

Mark 1.00 out of 1.00

A number is stable if each digit occur the same number of times.i.e, the frequency of each digit in the number is the same. For e.g. 2277,4004,11,23,583835,1010 are examples for stable numbers.
Similarly, a number is unstable if the frequency of each digit in the number is NOT same.

Sample Input:

2277

Sample Output:

Stable Number

Sample Input 2:

121

Sample Output 2:

Unstable Number

Answer: (penalty regime: 0 %)

```
def is_stable(number):

    number_str = str(number)


    digit_freq = {}
    for digit in number_str:
        if digit in digit_freq:
            digit_freq[digit] += 1
        else:
            digit_freq[digit] = 1


    freq_values = set(digit_freq.values())
    return len(freq_values) == 1


numbers = input().split()

for number in numbers:
    number = int(number)
    if is_stable(number):
```

	Input	Expected	Got	
✔	9988	Stable Number	Stable Number	✔
✔	12	Stable Number	Stable Number	✔
✔	455	Unstable Number	Unstable Number	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

Question **4**

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 %)

```
def count_distinct_numbers(arr, k):
    n = len(arr)
    distinct_count = {}


    for i in range(k):
        distinct_count[arr[i]] = distinct_count.get(arr[i], 0) + 1


    result = [len(distinct_count)]


    for i in range(k, n):

        if distinct_count[arr[i - k]] == 1:
            del distinct_count[arr[i - k]]
        else:
            distinct_count[arr[i - k]] -= 1


        distinct_count[arr[i]] = distinct_count.get(arr[i], 0) + 1
        result.append(len(distinct_count))
```

	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 **5**

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=input()
b=input()
c=input()
if c in a and b:
    print("yes")
    print("set3 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	✔

	Test	Input	Expected	Got	
✔	2	mango orange banana orange grapes	No	No	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

◄ Week-09_MCQ

Jump to...

WEEK-09-Extra ►