Started on	Friday, 3 May 2024, 1:02 PM
State	Finished
Completed on	Sunday, 5 May 2024, 10:56 AM
Time taken	1 day 21 hours
Marks	5.00/5.00
Grade	**50.00** out of 50.00 (**100**%)
Name	ISHWARYA M 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 %)

```
def return_list(str1):
   str1=str1.replace("{"," ")
   str1=str1.replace("}","")
   l=str1.split(",")
   list1=[]
   for ele in l:
     list1.append(int(ele))
   return list1
list1=input()
list2=input()
list3=input()
result=[]
list_of_list=[]

list_of_list.append(return_list(list1))
list_of_list.append(return_list(list2))

list_of_list.append(return_list(list3))

for j in list_of_list:
   for i in j:
     x=list_of_list[0].count(i)
     x+=list_of_list[1].count(i)
     x+=list_of_list[2].count(i)
     if x==1:
       result.append(i)
result.sort()
print("{",end="")
for i in range(len(result)-1):
 print(result[i],end=",")

print(result[-1],"}", sep="")

```

	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 anagram(a, b):
    a = a.lower()
    b = b.lower()
    if sorted(a) == sorted(b):
        print("Anagrams")
    else:
        print("Not Anagrams")

a = input()
b = input()
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):
    num_str = str(number)
    digit_count = {}
    for digit in num_str:
        if digit.isdigit():
            digit_count[digit] = digit_count.get(digit, 0) + 1
    frequency = None
    for count in digit_count.values():
        if frequency is None:
            frequency = count
        elif frequency != count:
            return False
    return True

number = int(input())
if is_stable_number(number):
    print("Stable Number")
else:
    print("Unstable 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

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 %)

```
def process_sentence(sentence):
    words = sentence.split()
    unique_words = sorted(set(words))
    unique_sentence = ' '.join(unique_words)
    count = sum(1 for word in unique_words if len(word) > 3)
    return unique_sentence, count

input_sentence = input()
unique_sentence, count = process_sentence(input_sentence)
print(unique_sentence)
print("Count = %d"%count)
```

	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 **5**

Correct

Mark 1.00 out of 1.00

Given a sorted linked list, delete all duplicates such that each element appear only *once*.

Example 1:

```
Input:
1 1 2
Output:
1 2
```

Example 2:

```
Input:
1 1 2 3 3
Output:
1 2 3
```

Answer: (penalty regime: 0 %)

```
arr = input().split()
list1 = set(map(int, arr))
list = sorted(list1)
for num in list:
    print(num, end=" ")

```

	Test	Input	Expected	Got	
✔	1	1 1 2	1 2	1 2	✔
✔	2	1 1 2 3 3	1 2 3	1 2 3	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

◄ Week-09_MCQ

Jump to...

WEEK-09-Extra ►