Started on	Monday, 29 April 2024, 4:17 PM
State	Finished
Completed on	Sunday, 5 May 2024, 9:00 PM
Time taken	6 days 4 hours
Marks	5.00/5.00
Grade	**50.00** out of 50.00 (**100**%)
Name	DHANUSH M 2022-CSD-A

Question **1**

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 areAnagrams(a, b):
    # Convert strings to lowercase
    a = a.lower()
    b = b.lower()

    # Create dictionaries to store character frequencies
    freq_a = {}
    freq_b = {}

    # Count frequencies of characters in string a
    for char in a:
        if char in freq_a:
            freq_a[char] += 1
        else:
            freq_a[char] = 1

    # Count frequencies of characters in string b
    for char in b:
        if char in freq_b:
```

```
            freq_b[char] += 1
        else:
            freq_b[char] = 1

    # Check if the two dictionaries are equal
    return freq_a == freq_b

# Example usage
def main():
    # Input
    a = input().strip()
    b = input().strip()

    # Check if strings are anagrams
    if areAnagrams(a, b):
        print("Anagrams")
    else:
        print("Not Anagrams")

# Test the function with the provided sample input
main()
```

	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 **2**

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 remove_duplicates_and_count_long_words(sentence):
  """
  This function removes duplicate words from a sentence, sorts them,
  prints the unique words, and counts words with length greater than 3.

  Args:
      sentence: The input sentence as a string.

  Returns:
      None
  """
  words = sentence.lower().split()
  unique_words = set(words)
  long_word_count = sum(len(word) > 3 for word in unique_words)
  print(" ".join(sorted(unique_words)))
  print("Count =", long_word_count)
sentence = input()
remove_duplicates_and_count_long_words(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 **3**

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 %)

```
def find_common_elements(list1, list2):
  """
  This function finds and prints common elements between two sorted lists.

  Args:
      list1: The first sorted list of integers.
      list2: The second sorted list of integers.

  Returns:
      None
  """
  i = j = 0
  common_elements = []

  while i < len(list1) and j < len(list2):
    if list1[i] == list2[j]:
      common_elements.append(list1[i])
      i += 1
      j += 1
    elif list1[i] < list2[j]:
      i += 1
    else:
      j += 1
  if common_elements:
    print(*common_elements, sep=" ")
  else:
    print("No common elements")
list1 = sorted(map(int, input().split()))
list2 = sorted(map(int, input().split()))
find_common_elements(list1, list2)
```

	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.

Question **4**

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 %)

```
set1 = set(input().split())
set2 = set(input().split())
set3 = set(input().split())

if set3.issubset(set1) and set3.issubset(set2):
    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	✔
✔	2	mango orange banana orange grapes	No	No	✔

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 %)

```
class ListNode:
    def __init__(self, val=0, next=None):
        self.val = val
        self.next = next
def deleteDuplicates(head):
    current = head
    while current and current.next:
        if current.val == current.next.val:
            current.next = current.next.next
        else:
            current = current.next
    return head
def printList(head):
    current = head
    while current:
        print(current.val, end=" ")
        current = current.next
    print()
def inputLinkedList():
    nums = list(map(int, input().split()))
    if not nums:
        return None
    dummy = ListNode()
    current = dummy
    for num in nums:
        current.next = ListNode(num)
        current = current.next
    return dummy.next
head = inputLinkedList()
head = deleteDuplicates(head)
printList(head)
```

	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 ►