Started on	Friday, 3 May 2024, 12:18 PM
State	Finished
Completed on	Friday, 17 May 2024, 1:18 PM
Time taken	14 days 1 hour
Marks	5.00/5.00
Grade	**50.00** out of 50.00 (**100**%)
Name	NAVEEN RAJ B 2022-CSD-A

Question **1**

Correct

Mark 1.00 out of 1.00

Mr.Harish is maintaining a phone directory which stores phone numbers. He will update the directory with phone numbers every week. While entering the input the number should not be stored inside if the phone number already exists. Finally he want his phone number to be printed in ascending order

Input: n – A1 array size and m – A2 arraysize

Array A1 containing phone numbers already existing and Array A2 containing numbers to be inserted

Ouput : Phone numbers printed in ascending order

Sample Test Case

Input

5

6

9840403212 9890909012 98123455 90123456 99123456

90909090 99999999 9840403212 12345678 12347890 99123456

Output

12345678 12347890 90123456 90909090 98123455 99123456 99999999 9840403212 9890909012

Answer: (penalty regime: 0 %)

```
def print_phone_numbers(A1, A2):
    phone_set = set(A1)
    for number in A2:
        if number not in phone_set:
            phone_set.add(number)
    sorted_numbers = sorted(phone_set, key=lambda x: int(x))  # Sorti
    print(" ".join(sorted_numbers))

# Sample Test Case
A1_size = int(input())
A2_size = int(input())

A1 = input().split()[:A1_size]
A2 = input().split()[:A2_size]

print_phone_numbers(A1, A2)

```

	Input	Expected	Got	
✔	3 3 9876543211 1122334455 6677889911 6677889911 9876543211 4455667788	1122334455 4455667788 6677889911 9876543211	1122334455 4455667788 6677889911 9876543211	✔

	Input	Expected	Got	
✔	5 6 9840403212 9890909012 98123455 90123456 99123456 90909090 99999999 9840403212 12345678 12347890 99123456	12345678 12347890 90123456 90909090 98123455 99123456 99999999 9840403212 9890909012	12345678 12347890 90123456 90909090 98123455 99123456 99999999 9840403212 9890909012	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

Question **2**

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: ListNode) -> ListNode:
    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 createLinkedListFromInput() -> ListNode:
    nums = list(map(int, input().split()))
    dummy = ListNode()
    current = dummy
    for num in nums:
        current.next = ListNode(num)
        current = current.next
    return dummy.next
```

	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.

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):

    number_str = str(number)
    digit_count = {}
    for digit in number_str:
        if digit in digit_count:
            digit_count[digit] += 1
        else:
            digit_count[digit] = 1
    frequency_values = set(digit_count.values())
    if len(frequency_values) == 1:
        return "Stable Number"
    else:
        return "Unstable Number"
user_input = input()


try:
    number = int(user_input)
except ValueError:
    print()
    exit()
```

	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

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 %)

```
s1 = input()
s2 = input()
s3 = input()

l1 = s1.split()
l2 = s2.split()

if s3 in l1 and l2:
    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

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_in_windows(arr, K):
    result = []
    window_set = set()
    for i in range(len(arr) - K + 1):
        window_set.clear()
        for j in range(i, i + K):
            window_set.add(arr[j])
        result.append(len(window_set))
    return result

# Example usage
arr = [1, 2, 1, 3, 4, 3]
K = 3
output = count_distinct_in_windows(arr, K)
for count in output:
    print(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.

◄ Week-09_MCQ

Jump to...

WEEK-09-Extra ►