Started on	Sunday, 19 May 2024, 8:43 PM
State	Finished
Completed on	Sunday, 19 May 2024, 9:40 PM
Time taken	56 mins 36 secs
Marks	7.00/7.00
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

To Check if a Given Key Exists in a Dictionary or Not

Input: Any dictionary format input (Ex: d={'A':1,'B':2,'C':3})

Enter Key to check: A

Output:

Key is present and value of the key is: (location)

Present # True Statement

Not Present # False Statement

Answer: (penalty regime: 0 %)

```
d = {'A':1, 'B':2, 'C':3}
a =input()
if a in d:
    print("Present")
else:
    print("Not Present")
```

	Input	Expected	Got	
✔	A	Present	Present	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

Question **2**

Correct

Mark 1.00 out of 1.00

A teacher wants to evaluate her class results for the subject she handles. She want to do the following analysis:

1. Display Class average
2. Display Maximum mark Roll no
3. Display Minimum mark Roll no

Kindly help her out. Use dictionary for storing the student details.

Input Format:

In line 1 no of students will be given

Followed by n lines containing student rollno and marks

Output Format:

Line 1 Class average

Line 2 Maximum mark Roll no

Line 3 Minimum mark Roll no

Sample Input:

4

01 87

02 99

03 45

04 77

Output:

77

02

03

Answer: (penalty regime: 0 %)

```
def evaluate_class_results():
    import sys
    input = sys.stdin.read

    data = input().strip().split()

    n = int(data[0])  # number of students

    # Initialize dictionary to store student roll number and marks
    student_marks = {}

    # Read student details
    for i in range(1, len(data), 2):
        roll_no = data[i]
        marks = int(data[i + 1])
        student_marks[roll_no] = marks

    # Calculate class average
    total_marks = sum(student_marks.values())
    class_average = total_marks / n

```

```
22    # Find the roll number with maximum and minimum marks
```

	Input	Expected	Got	
✔	4 01 87 02 99 03 45 04 77	77 02 03	77 02 03	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

Question **3**

Correct

Mark 1.00 out of 1.00

Multiply All the Items in a Dictionary

Input: Any input in Dictionary format (Ex: d={'A':10,'B':10,'C':239})

Output: multiplication of dictionary values (23900)

Answer: (penalty regime: 0 %)

```
def multiply_dict_values(d):
    result = 1
    for value in d.values():
        result *= value
    return result

# Example usage:
d = {'A': 10, 'B': 10, 'C': 239}
result = multiply_dict_values(d)
print(result)  # Output: 23900
```

	Input	Expected	Got	
✔	d={'A':10,'B':10,'C':239}	23900	23900	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

Question **4**

Correct

Mark 1.00 out of 1.00

A sentence is a list of words that are separated by a single space with no leading or trailing spaces. Each word consists of lowercase and uppercase English letters.

A sentence can be shuffled by appending the 1-indexed word position to each word then rearranging the words in the sentence.

For example, the sentence "This is a sentence" can be shuffled as "sentence4 a3 is2 This1" or "is2 sentence4 This1 a3".

Given a shuffled sentence s containing no more than 9 words, reconstruct and return the original sentence.

Example 1:

Input:

is2 sentence4 This1 a3

Output:

This is a sentence

Explanation: Sort the words in s to their original positions "This1 is2 a3 sentence4", then remove the numbers.

Example 2:

Input:

Myself2 Me1 I4 and3

Output:

Me Myself and I

Explanation: Sort the words in s to their original positions "Me1 Myself2 and3 I4", then remove the numbers.

Constraints:

2 <= s.length <= 200

s consists of lowercase and uppercase English letters, spaces, and digits from 1 to 9.

The number of words in s is between 1 and 9.

The words in s are separated by a single space.

s contains no leading or trailing spaces.

Answer: (penalty regime: 0 %)

```
def reconstruct_sentence(s: str) -> str:
    # Split the shuffled sentence into words
    words = s.split()

    # Sort the words based on the numeric suffix
    sorted_words = sorted(words, key=lambda word: int(word[-1]))

    # Remove the numeric suffix and join the words back into a sentence
    original_sentence = ' '.join(word[:-1] for word in sorted_words)

    return original_sentence

# Get input from the user
shuffled_sentence = input("")

# Reconstruct the original sentence and print it
original_sentence = reconstruct_sentence(shuffled_sentence)
```

```
18 print(original_sentence)
```

	Input	Expected	Got	
✔	is2 sentence4 This1 a3	This is a sentence	This is a sentence	✔
✔	Myself2 Me1 Vijay4 and3	Me Myself and Vijay	Me Myself and Vijay	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

Question **5**

Correct

Mark 1.00 out of 1.00

In the game of Scrabble™, each letter has points associated with it. The total score of a word is the sum of the scores of its letters. More common letters are worth fewer points while less common letters are worth more points. The points associated with each letter are shown below:

Points Letters

1 A, E, I, L, N, O, R, S, T and U

2 D and G

3 B, C, M and P

4 F, H, V, W and Y

5 K

8 J and X

10 Q and Z

Write a program that computes and displays the Scrabble™ score for a word. Create a dictionary that maps from letters to point values. Then use the dictionary to compute the score.

A Scrabble™ board includes some squares that multiply the value of a letter or the value of an entire word. We will ignore these squares in this exercise.

Sample Input

REC

Sample Output

REC is worth 5 points.

Answer: (penalty regime: 0 %)

```
def scrabble_score(word):
    # Dictionary mapping letters to their Scrabble point values
    letter_points = {
        'A': 1, 'E': 1, 'I': 1, 'L': 1, 'N': 1, 'O': 1, 'R': 1, 'S': 1, 'T':
        'D': 2, 'G': 2,
        'B': 3, 'C': 3, 'M': 3, 'P': 3,
        'F': 4, 'H': 4, 'V': 4, 'W': 4, 'Y': 4,
        'K': 5,
        'J': 8, 'X': 8,
        'Q': 10, 'Z': 10
    }

    # Initialize the total score to 0
    total_score = 0

    # Convert the word to uppercase to match the dictionary keys
    word = word.upper()

    # Compute the total score by summing up the points of each letter in the
    for letter in word:
        if letter in letter_points:
            total score += letter points[letter]
```

	Input	Expected	Got	
✔	REC	REC is worth 5 points.	REC is worth 5 points.	✔
✔	RAJALAKSHMI	RAJALAKSHMI is worth 27 points.	RAJALAKSHMI is worth 27 points.	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

Question **6**

Correct

Mark 1.00 out of 1.00

Create a program that determines and displays the number of unique characters in a string entered by the user. For example, Hello, World! has 10 unique characters while zzz has only one unique character. Use a dictionary or set to solve this problem.

For example:

Input	Result
Hello, World!	10

Answer: (penalty regime: 0 %)

```
def count_unique_characters(input_string):
    # Use a set to store unique characters
    unique_characters = set(input_string)

    # Return the number of unique characters
    return len(unique_characters)

# Get input from the user
user_input = input("")

# Calculate the number of unique characters
unique_count = count_unique_characters(user_input)

# Display the result
print( unique_count)
```

	Input	Expected	Got	
✔	Hello, World!	10	10	✔
✔	zzz	1	1	✔
✔	RECCSE	4	4	✔
✔	AAABBBCCC	3	3	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

Question **7**

Correct

Mark 1.00 out of 1.00

Two words are anagrams if they contain all of the same letters, but in a different order. For example, “evil” and “live” are anagrams because each contains one “e”, one “i”, one “l”, and one “v”. Create a program that reads two strings from the user, determines whether or not they are anagrams, and reports the result.

Sample Input 1

evil

live

Sample Output 1

Those strings are anagrams.

Sample Input 2

meet

met

Sample Output 2

Those strings are not anagrams.

Answer: (penalty regime: 0 %)

```
def are_anagrams(str1, str2):
    # Convert strings to lowercase and remove spaces
    str1 = str1.lower().replace(" ", "")
    str2 = str2.lower().replace(" ", "")

    # Check if the lengths of the strings are equal
    if len(str1) != len(str2):
        return False

    # Sort the characters in both strings and compare
    return sorted(str1) == sorted(str2)

# Input
string1 = input()
string2 = input()

# Check if strings are anagrams
if are_anagrams(string1, string2):
    print("Those strings are anagrams.")
else:
    print("Those strings are not anagrams.")
```

	Input	Expected	Got	
✔	evil live	Those strings are anagrams.	Those strings are anagrams.	✔
✔	meet met	Those strings are not anagrams.	Those strings are not anagrams.	✔
✔	rec cer	Those strings are anagrams.	Those strings are anagrams.	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

◄ Week-10_MCQ

Jump to...

WEEK-10-Extra ►