Submitted By : Sardar Muhammad Saad

Class : BSSE3rd

Section : C

Roll no : 12386

LabTask : 06

Submitted to : Sir Jammal Abdul Ahad

Date : 26-Nov-2023

Q#01)

Create a function in python that takes two strings as input and determines if they are anagrams. Utilize a hash table to efficiently compare the character frequencies in two strings.

Ans#01)

```
def are_anagrams(str1, str2):
    # Removing spaces and converting to lowercase for case-insensitivity
    str1 = str1.replace(" ", "").lower()
    str2 = str2.replace(" ", "").lower()

    # Check if lengths are equal after removing spaces
    if len(str1) != len(str2):
        return False
```

```
    # Use dictionaries to store character frequencies
    char_freq1 = {}
    char_freq2 = {}

    # Count character frequencies in the first string
    for char in str1:
        char_freq1[char] = char_freq1.get(char, 0) + 1

    # Count character frequencies in the second string
    for char in str2:
        char_freq2[char] = char_freq2.get(char, 0) + 1

    # Compare the dictionaries
    return char_freq1 == char_freq2

# Example usage:
string1 = "listen"
string2 = "silent"
result = are_anagrams(string1, string2)
print(f"{string1} and {string2} are anagrams: {result}")
```

Input for the program (Optional)

Output:

```
listen and silent are anagrams: True
```

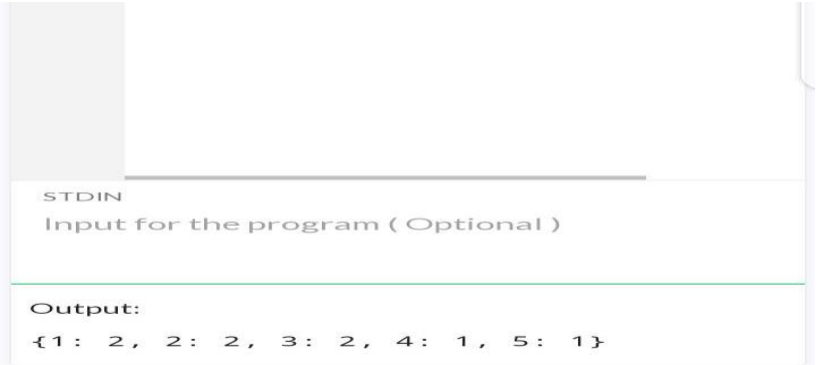
Q#02)

Write a Python function that takes a list of Integer as input and returns a dictionary Where keys are unique number from the list, And values are the frequencies of those numbers.

Ans#02)

```
def count_frequencies(numbers):
    frequency_dict = {}
    for num in numbers:
        if num in frequency_dict:
            frequency_dict[num] += 1
        else:
            frequency_dict[num] = 1
    return frequency_dict

numbers_list = [1, 2, 3, 1, 2, 3, 4, 5]
result_dict = count_frequencies(numbers_list)
print(result_dict)
```



Q#03)

Implement a Python function that, Given an array of integers, Find the length of the longest subarray With the sum equal to specified value K. Use a hash table to track cumulative sums effectively

Ans#03)

```
def longest_subarray_with_sum(arr, K):
    cum_sum = 0  # Initialize cumulative sum
    max_length = 0  # Initialize the maximum length of subarray
    sum_index = {}  # Create a hash table to store cumulative sums and their indices

    for i in range(len(arr)):
        cum_sum += arr[i]

        # Check if cumulative sum is equal to K
        if cum_sum == K:
            max_length = i + 1  # Update max length

        # If cumulative sum - K is already in the hash table, update max length
        if cum_sum - K in sum_index:
            max_length = max(max_length, i - sum_index[cum_sum - K])

        # If cumulative sum is not in the hash table, add it with its index
```

```
        if cum_sum not in sum_index:
            sum_index[cum_sum] = i

    return max_length

# Example usage:
arr = [10, 5, 2, 7, 1, 9]
K = 15
result = longest_subarray_with_sum(arr, K)
print("Length of the longest subarray with sum", K, "is:", result)
```