ABBOTTABAD UNIVERSITY OF SCIENCE AND TECHNOLOGY

NAME	**SAAD SHAH**
ROLL NO	**12373**
SUBJECT	**DSA**
LAB TASKS	**06**
SUBMITTED TO	**MR JAMAL ABDUL AHAD**
DATED	**11-19-2023**

Q NO 1 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 the two strings.

```python
def are_anagrams(str1, str2):
    # Remove spaces and convert strings to lowercase
    str1 = str1.replace(' ', '').lower()
    str2 = str2.replace(' ', '').lower()

    # If lengths of strings are different, they can't be anagrams
    if len(str1) != len(str2):
        return False

    # Create a dictionary to store character frequencies of str1
    char_freq = {}

    # Count frequencies of characters in str1
    for char in str1:
        if char in char_freq:
            char_freq[char] += 1
        else:
            char_freq[char] = 1

    # Decrement frequencies for characters in str2
    for char in str2:
        if char in char_freq:
            char_freq[char] -= 1
        else:
            # If a character in str2 is not in the dictionary, they can't be anagrams
            return False

    # Check if all character frequencies are 0, indicating anagrams
    for value in char_freq.values():
        if value != 0:
            return False

    # If all character frequencies are 0, strings are anagrams
    return True

# Example usage:
string1 = "listen"
string2 = "silent"
if are_anagrams(string1, string2):
```

PROBLEMS   OUTPUT   DEBUG CONSOLE   **TERMINAL**   PORTS

```
PS C:\Users\hp\Desktop\python\codes> python -u "c:\Users\hp\Desktop\python\codes\tempCodeRunnerFile.python"
'listen' and 'silent' are anagrams.
PS C:\Users\hp\Desktop\python\codes>
```

Q NO 2: Write a Python function that takes a list of integers as input and returns a dictionary where keys are unique numbers from the list, and values are the frequencies of those numbers.

```python
def count_frequency(nums):
    frequency_dict = {}

    for num in nums:
        if num in frequency_dict:
            frequency_dict[num] += 1
        else:
            frequency_dict[num] = 1

    return frequency_dict

# Example usage:
numbers = [1, 2, 3, 4, 2, 3, 1, 2, 4, 5, 6, 5]
result = count_frequency(numbers)
print("Number frequencies:", result)
```

PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS   Code + ∨

```
PS C:\Users\hp\Desktop\python\codes> python -u "c:\Users\hp\Desktop\python\codes\tempCodeRunnerFile.python"
Number frequencies: {1: 2, 2: 3, 3: 2, 4: 2, 5: 2, 6: 1}
PS C:\Users\hp\Desktop\python\codes>
```

Q NO 3 : Implement a Python function that, given an array of integers, finds the length of the longest subarray with a sum equal to a specified value K. Use a hash table to track cumulative sums efficiently.

```python
def longest_subarray_with_sum(nums, k):
    max_length = 0
    cum_sum = 0
    sum_indices = {0: -1}  # Initialize with 0 cumulative sum at index -1

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

        # Check if (cumulative sum - k) exists in the dictionary
        if cum_sum - k in sum_indices:
            max_length = max(max_length, i - sum_indices[cum_sum - k])

        # Store the cumulative sum if not already in the dictionary
        if cum_sum not in sum_indices:
            sum_indices[cum_sum] = i

    return max_length

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

PROBLEMS   OUTPUT   DEBUG CONSOLE   **TERMINAL**   PORTS   >_ Code + ∨

```
PS C:\Users\hp\Desktop\python\codes> python -u "c:\Users\hp\Desktop\python\codes\tempCodeRunnerFile.python"
Length of the longest subarray with sum 15 is 4
PS C:\Users\hp\Desktop\python\codes>
```