

**ABBOTTABAD UNIVERSITY OF SCIENCE AND TECHNOLOGY ABBOTTABAD**

**NAME : Sayyed Zain Ul Abidin Shah**

**SUBMITTED TO: Sir Abdul Ahad**

**ROLL NO : 12387**

**SUBJECT : DSA**

**DATE OF SUBMISSION : 26 Nov. 23**

**Q1: Create a function in Python that takes two strings as input and**

**Determines if they are anagrams. Utilize a hash table too efficiently**

**Compare the character frequencies in the two strings.**

**def are\_anagrams(str1, str2):**

**# Remove spaces and convert to lowercase for case-insensitive comparison**

**str1 = str1.replace(" ", "").lower()**

**str2 = str2.replace(" ", "").lower()**

**# Check if lengths are different**

**if len(str1) != len(str2):**

**return False**

**# Create hash tables to store character frequencies**

**freq\_table1 = {}**

**freq\_table2 = {}**

**# Populate the first frequency table**

**for char in str1:**

**freq\_table1[char] = freq\_table1.get(char, 0) + 1**

**# Populate the second frequency table**

**for char in str2:**

**freq\_table2[char] = freq\_table2.get(char, 0) + 1**

**# Compare the two frequency tables**

**return freq\_table1 == freq\_table2**

**# Example usage:**

**string1 = "listen"**

**string2 = "silent"**

**result = are\_anagrams(string1, string2)**

**print(result) # Output: True**

**Q2: 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 ofthose numbers.**

**def count\_frequencies(numbers):**

**frequency\_dict = {}**

**for num in numbers:**

**frequency\_dict[num] = frequency\_dict.get(num, 0) + 1**

**return frequency\_dict**

**# Example usage:**

**numbers\_list = [1, 2, 3, 1, 2, 3, 4, 5]**

**result\_dict = count\_frequencies(numbers\_list)**

**print(result\_dict)**

**# Output: {1: 2, 2: 2, 3: 2, 4: 1, 5: 1}**

**Q3: 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.**

**def longest\_subarray\_with\_sum(arr, k):**

**cumulative\_sum = 0**

**max\_length = 0**

**sum\_indices = {0: -1} # Initialize with a dummy sum 0 at index -1**

**for i, num in enumerate(arr):**

**cumulative\_sum += num**

**if cumulative\_sum - k in sum\_indices:**

**max\_length = max(max\_length, i - sum\_indices[cumulative\_sum - k])**

**if cumulative\_sum not in sum\_indices:**

**sum\_indices[cumulative\_sum] = i**

**return max\_length**

**# Example usage:**

**array = [10, 5, 2, 7, 1, 9]**

**target\_sum = 15**

**result = longest\_subarray\_with\_sum(array, target\_sum)**

**print(result) # Output: 4 (subarray: [5, 2, 7, 1])**