# String Manipulation Problems

## Problem 1: Length of the Longest Word

* **Description**: Find the length of the longest word in a given string. Words are separated by spaces. Do not use built-in string manipulation functions.
* **Example Input**: ”This is a sample sentence.”
* **Example Output**: 7 (the word ”sentence”)

### Code:

def longest\_word\_length(s): s\_list = s.split()

if not s\_list: return 0

else:

d = {i: len(i) for i in s\_list} s\_values = [v for k, v in d.items()] list\_values = list(s\_values)

return max(list\_values)

1

2

3

4

5

6

7

8

9

## Problem 2: Check for Substring

* **Description**: Determine if string t is a substring of string s. Do not use built-in string functions or recursion.
* **Example Input**: s = ”This is a test string”, t = ”test”
* **Example Output**: True

### Code:

def is\_substring(s, t): n\_s = len(s)

n\_t = len(t)

for i in range(n\_s - n\_t + 1): if s[i:i + n\_t] == t:

return True return False

1

2

3

4

5

6

7

## Problem 3: Check Subsequence

* **Description**: Determine if string t is a subsequence of string s. A sub- sequence is a sequence derived by deleting some or no elements without changing the order of the remaining elements.
* **Example Input**: s = ”abcde”, t = ”ace”
* **Example Output**: True

### Code:

def is\_subsequence(s, t): i, j = 0, 0

while i < len(s) and j < len(t): if s[i] == t[j]:

j += 1

i += 1

return j == len(t)

1

2

3

4

5

6

7

## Problem 4: Check for Anagrams

* **Description**: Determine if string t is an anagram of string s. An anagram is a word or phrase formed by rearranging the letters of another.
* **Example Input**: s = ”listen”, t = ”silent”
* **Example Output**: True

### Code:

def is\_anagram(s, t):

if sorted(s.lower()) == sorted(t.lower()): return True

else:

return False

1

2

3

4

5

## Problem 5: Count Consonants in a String

* **Description**: Count the number of consonants in a given string.
* **Example Input**: ”Hello World”

### Example Output: 7

* **Code**:

1

def count\_consonants(s): vowels = ’aeiouAEIOU’ count = 0

for char in s:

if (’a’ <= char <= ’z’) or (’A’ <= char <= ’Z’): if char not in vowels:

count += 1 return count

2

3

4

5

6

7

8

## Problem 6: Remove Duplicates in a String

* **Description**: Remove duplicate characters from a string while preserving the order of the first occurrences.
* **Example Input**: ”programming”
* **Example Output**: ”progamin”

### Code:

def remove\_duplicates(s): unique = ””

for i in s:

if i not in unique: unique += i

else:

pass return unique

1

2

3

4

5

6

7

8

## Problem 7: Count Words in a String

* **Description**: Count the number of words in a given string (words are sep- arated by spaces).
* **Example Input**: ”This is a sentence.”

### Example Output: 4

* **Code**:

def count\_words(s): s1 = s.split() n = len(s1) return n

1

2

3

4

## Problem 8: Check for Same Strings

* **Description**: Check if two strings are equal (same length and characters at each position). Do not use built-in string comparison functions.
* **Example Input**: s = ”hello”, t = ”hello”
* **Example Output**: True

### Code:

def are\_equal\_strings(s, t):

if len(s) == len(t) and s == t: return True

else:

return False

1

2

3

4

5

## Problem 9: Check Palindrome

* **Description**: Check if a string is a palindrome (reads the same forwards and backward, ignoring case and non-alphanumeric characters).
* **Example Input**: ”A man, a plan, a canal: Panama”
* **Example Output**: True

### Code:

import re

def is\_palindrome(s):

s = re.sub(r’[^a-zA-Z0-9]’, ’’, s).lower() return s == s[::-1]

1

2

3

4

5

## Problem 10: Count Vowels in a String

* **Description**: Count the number of vowels (both uppercase and lowercase) in a given string.
* **Example Input**: ”Hello World”

### Example Output: 3

* **Code**:

def count\_vowels(s): a = ’aeiouAEIOU’ count = 0

for i in s:

if i in a:

1

2

3

4

5

6

count += 1 return count

7

# Programming Problems

## Problem 1: Reverse a String

* **Description**: Reverse a given string.
* **Example Input**: ”hello”
* **Example Output**: ”olleh”

### Code:

def reverse\_string(s): return s[::-1]

1

2

## Problem 2: Binary to Decimal

* **Description**: Convert a binary string to its decimal equivalent.
* **Example Input**: ”1011”
* **Example Output**: 11

### Code:

def binary\_to\_decimal(binary\_str): decimal = 0

power = 0

for digit in reversed(binary\_str): if digit == ’1’:

decimal += 2\*\*power power += 1

return decimal

1

2

3

4

5

6

7

8

## Problem 3: Decimal to Binary

* **Description**: Convert a decimal integer to its binary representation (as a string). Handle negative numbers.
* **Example Input**: 13, -13
* **Example Output**: ”1101”, ”-1101”

### Code:

def int\_to\_binary(n): if n == 0:

return ”0” negative = n < 0 n = abs(n) binary = ”” while n > 0:

binary = str(n % 2) + binary n //= 2

return ”-” + binary if negative else binary

1

2

3

4

5

6

7

8

9

10

## Problem 4: Valid Perfect Square

* **Description**: Check if a positive integer is a perfect square.
* **Example Input**: 16, 14
* **Example Output**: True, False

### Code:

def is\_perfect\_square(num): if num < 1:

return False a = 1

while a \* a <= num: if a \* a == num:

return True a += 1

return False

1

2

3

4

5

6

7

8

9

## Problem 5: Check for Prime Number

* **Description**: Check if a number is a prime number.
* **Example Input**: 17, 15
* **Example Output**: True, False

### Code:

def is\_prime(n): if n <= 1:

return False

for i in range(2, int(n\*\*0.5) + 1): if n % i == 0:

1

2

3

4

5

6

return False return True

7

## Problem 6: Check for Even Number

* **Description**: Check if a number is even.
* **Example Input**: 12, 7
* **Example Output**: True, False

### Code:

def is\_even(n): return n % 2 == 0

1

2

## Problem 7: Sum of N Even Natural Numbers

* **Description**: Calculate the sum of the first n even natural numbers.

### Example Input: 3

* **Example Output**: 12 (2 + 4 + 6)

### Code:

def sum\_of\_even\_numbers(n): return n \* (n + 1)

1

2

## Problem 8: Check if List is Subset of another List

* **Description**: Check if one list is a subset of another list (brute-force ap- proach, without using the in keyword).
* **Example Input**: lst1 = [1, 2, 3], lst2 = [1, 2, 3, 4, 5], lst1

= [1, 6], lst2 = [1, 2, 3, 4, 5]

* **Example Output**: True, False

### Code:

def is\_subset(lst1, lst2): for element in lst1:

found = False

for item in lst2:

if item == element: found = True break

1

2

3

4

5

6

7

8

if not found:

return False return True

9

10

## Problem 9: Merge Dictionaries with Common Keys

* **Description**: Merge multiple dictionaries, summing values for common keys.
* **Example Input**: [’a’: 1, ’b’: 2, ’b’: 3, ’c’: 4, ’c’: 5, ’d’:

6]

* **Example Output**: ’a’: 1, ’b’: 5, ’c’: 9, ’d’: 6

### Code:

def merge\_dicts\_with\_overlapping\_keys(dicts): merged = {}

for d in dicts:

for k, v in d.items():

merged[k] = merged.get(k, 0) + v return merged

1

2

3

4

5

6

## Problem 10: Palindromic Tuple

* **Description**: Check if a tuple is a palindrome (reads the same forwards and backward).
* **Example Input**: (1, 2, 3, 2, 1), (1, 2, 3, 4, 5)
* **Example Output**: True, False

### Code:

def is\_palindromic\_tuple(tup):

return list(tup) == list(tup)[::-1]

1

2

# Programming Problems

## Problem 1: Word Frequency in a Sentence

* **Description**: Count the frequency of words in a sentence and store them in a dictionary.
* **Condition**: The sentence is converted to lowercase for case-insensitive count- ing, and words are separated by spaces.
* **Example Input**: ”the quick brown fox jumps over the lazy dog”
* **Example Output**: {’the’: 2, ’quick’: 1, ’brown’: 1, ’fox’: 1,

’jumps’: 1, ’over’: 1, ’lazy’: 1, ’dog’: 1}

### Code:

from collections import Counter

def count\_word\_frequency(sentence): words = sentence.lower().split() return dict(Counter(words))

1

2

3

4

5

## Problem 2: Merge Three Dictionaries

* **Description**: Merge three dictionaries into a single dictionary. If keys are duplicated, the last dictionary’s value for that key will overwrite earlier ones.
* **Condition**: The merging process copies the first dictionary and updates it with the second and third dictionaries.
* **Example Input**: dict1 = {’a’: 1, ’b’: 2}, dict2 = {’c’: 3, ’d’: 4}, dict3 = {’e’: 5, ’f’: 6}
* **Example Output**: {’a’: 1, ’b’: 2, ’c’: 3, ’d’: 4, ’e’: 5, ’f’:

6}

### Code:

1

def merge\_three\_dictionaries(dict1, dict2, dict3): merged = dict1.copy()

merged.update(dict2) merged.update(dict3) return merged

2

3

4

5

## Problem 3: Merge Lists to Dictionary

* **Description**: Merge two lists into a dictionary where the first list’s elements are keys and the second list’s elements are values. Handle cases where lists have unequal lengths.
* **Condition**: If the lists have unequal lengths, return False. Otherwise, pair corresponding elements as key-value pairs.
* **Example Input 1**: keys = [’a’, ’b’, ’c’], values = [1, 2, 3]
* **Example Output 1**: {’a’: 1, ’b’: 2, ’c’: 3}
* **Example Input 2**: keys = [’x’, ’y’], values = [10, 20, 30]
* **Example Output 2**: False

### Code:

def merge\_lists\_to\_dictionary(keys, values): if len(keys) != len(values):

return False

return dict(zip(keys, values))

1

2

3

4

## Problem 4: Rotate a List

* **Description**: Rotate a list of integers to the right by k positions without using slicing.
* **Condition**: Handle empty lists and cases where k is larger than the list length by using modulo. Rotation is performed by repeatedly moving the last element to the front.
* **Example Input 1**: lst = [1, 2, 3, 4, 5], k = 2
* **Example Output 1**: [4, 5, 1, 2, 3]
* **Example Input 2**: lst = [10, 20, 30, 40, 50], k = 3
* **Example Output 2**: [30, 40, 50, 10, 20]

### Code:

1

def rotate\_list(lst, k): n = len(lst)

if not lst:

return [] k %= n

for \_ in range(k): last\_element = lst.pop() lst.insert(0, last\_element)

return lst

2

3

4

5

6

7

8

9

## Problem 5: Count Vowels in a String

* **Description**: Count the number of vowels (both uppercase and lowercase) in a string.
* **Condition**: Vowels are defined as ’a’, ’e’, ’i’, ’o’, ’u’ (case-insensitive).
* **Example Input**: ”Hello World”

### Example Output: 3

* **Code**:

def count\_vowels(s): vowels = ”aeiouAEIOU”

count = sum(1 for char in s if char in vowels) return count

1

2

3

4

# Programming Problems

## Problem 1: Palindromic Tuple

* **Description**: Check if a tuple is a palindrome (reads the same forwards and backward).
* **Condition**: Convert the tuple to a list and compare it with its reverse.
* **Example Input 1**: (1, 2, 3, 2, 1)
* **Example Output 1**: True
* **Example Input 2**: (1, 2, 3, 4, 5)
* **Example Output 2**: False

### Code:

def is\_palindromic\_tuple(tup):

return list(tup) == list(tup)[::-1]

1

2

## Problem 2: Word Frequency in a Sentence

* **Description**: Count word frequencies in a sentence.
* **Condition**: Convert the sentence to lowercase for case-insensitive counting and use Counter for efficiency.
* **Example Input**: ”the quick brown fox jumps over the lazy dog”
* **Example Output**: {’the’: 2, ’quick’: 1, ’brown’: 1, ’fox’: 1,

’jumps’: 1, ’over’: 1, ’lazy’: 1, ’dog’: 1}

### Code:

from collections import Counter

def count\_word\_frequency(sentence): words = sentence.lower().split() return dict(Counter(words))

1

2

3

4

5

## Problem 3: Merge Three Dictionaries

* **Description**: Merge three dictionaries; later dictionaries overwrite earlier ones if keys conflict.
* **Condition**: Copy the first dictionary and update it with the second and third dictionaries.
* **Example Input**: dict1={’a’:1,’b’:2}, dict2={’c’:3,’d’:4}, dict3={’e’:5,
* **Example Output**: {’a’: 1, ’b’: 2, ’c’: 3, ’d’: 4, ’e’: 5, ’f’:

6}

### Code:

def merge\_three\_dictionaries(dict1, dict2, dict3): merged = dict1.copy()

merged.update(dict2) merged.update(dict3) return merged

1

2

3

4

5

## Problem 4: Merge Lists to Dictionary

* **Description**: Create a dictionary from two lists, using the first list as keys and the second as values. Handle unequal list lengths.
* **Condition**: Return False if lists have unequal lengths; otherwise, pair el- ements using zip.
* **Example Input 1**: keys=[’a’,’b’,’c’], values=[1,2,3]
* **Example Output 1**: {’a’: 1, ’b’: 2, ’c’: 3}
* **Example Input 2**: keys=[’x’,’y’], values=[10,20,30]
* **Example Output 2**: False

### Code:

def merge\_lists\_to\_dictionary(keys, values): if len(keys) != len(values):

return False

return dict(zip(keys, values))

1

2

3

4

## Problem 5: Rotate a List

* **Description**: Rotate a list to the right by k positions without using slicing.
* **Condition**: Handle empty lists and k larger than list length using modulo. Rotate by moving the last element to the front k times.
* **Example Input**: lst=[1,2,3,4,5], k=2
* **Example Output**: [4, 5, 1, 2, 3]

### Code:

def rotate\_list(lst, k): n = len(lst)

if not lst:

return [] k %= n

for \_ in range(k): lst.insert(0, lst.pop())

return lst

1

2

3

4

5

6

7

8

## Problem 6: Merge Two Sorted Lists

* **Description**: Merge two sorted lists into a single sorted list.
* **Condition**: Combine the lists and use sorted for efficiency.
* **Example Input**: list1=[1,3,5], list2=[2,4,6]
* **Example Output**: [1, 2, 3, 4, 5, 6]

### Code:

def merge\_two\_sorted\_lists(list1, list2): merged = sorted(list1 + list2) return merged

1

2

3

## Problem 7: Maximum Difference Between Two Con- secutive Elements

* **Description**: Find the maximum difference between consecutive elements in a list.
* **Condition**: Return 0 for lists with fewer than 2 elements. Compute absolute differences between consecutive pairs.
* **Example Input**: lst=[1,7,3,10,5]

### Example Output: 7

* **Code**:

def max\_consecutive\_difference(lst): if len(lst) < 2:

return 0

max\_diff = 0

1

2

3

4

5

for i in range(len(lst) - 1): diff = abs(lst[i] - lst[i+1]) max\_diff = max(max\_diff, diff)

return max\_diff

6

7

8

## Problem 8: Count Number of Odd and Even Elements

* **Description**: Count the number of even and odd numbers in a list.
* **Condition**: Count even numbers directly and compute odd count as the difference from total length.
* **Example Input**: lst=[1,2,3,4,5]
* **Example Output**: (2, 3) (2 even, 3 odd)

### Code:

def count\_even\_odd(lst):

even\_count = sum(1 for num in lst if num % 2 == 0) odd\_count = len(lst) - even\_count

return (even\_count, odd\_count)

1

2

3

4

## Problem 9: Reverse a List

* **Description**: Reverse a list without using slicing.
* **Condition**: Build a new list by appending elements in reverse order.
* **Example Input**: lst=[1,2,3,4,5]
* **Example Output**: [5, 4, 3, 2, 1]

### Code:

def reverse\_list(lst): reversed\_lst = []

for i in range(len(lst) - 1, -1, -1): reversed\_lst.append(lst[i])

return reversed\_lst

1

2

3

4

5

## Problem 10: Check if All Elements in a List are Unique

* **Description**: Check if all elements in a list are unique.
* **Condition**: Compare the length of the list with the length of its set.
* **Example Input 1**: lst=[1,2,3,4,5]
* **Example Output 1**: True
* **Example Input 2**: lst=[1,2,3,3,4,5]
* **Example Output 2**: False

### Code:

def check\_unique(lst):

return len(lst) == len(set(lst))

1

2

## Problem 11: Remove Duplicates from a List

* **Description**: Remove duplicates from a list while maintaining order.
* **Condition**: Build a new list, adding each element only if it hasn’t been added before.
* **Example Input**: lst=[1,2,2,3,4,4,5]
* **Example Output**: [1, 2, 3, 4, 5]

### Code:

def remove\_duplicates(lst): unique\_lst = []

for item in lst:

if item not in unique\_lst: unique\_lst.append(item)

return unique\_lst

1

2

3

4

5

6

## Problem 12: Find the Largest Element in a List

* **Description**: Find the largest element in a list.
* **Condition**: Use the built-in max function for efficiency.
* **Example Input**: numbers=[3, 8, 2, 10, 5]
* **Example Output**: 10

### Code:

def find\_largest(numbers): return max(numbers)

1

2

## Problem 13: Sum of List Elements

* **Description**: Calculate the sum of elements in a list.
* **Condition**: Use the built-in sum function for efficiency.
* **Example Input**: numbers=[1,2,3,4,5]
* **Example Output**: 15

### Code:

def sum\_list(numbers): return sum(numbers)

1

2