ANALYTICS WITH RAHUL

Python Practice Problems

Get Expertise in coding

Problem Statement – Disarium Number

A **Disarium Number** is a number in which the sum of its digits powered with their respective positions is equal to the number itself. The digits are indexed starting from 1. For example, 175 is a Disarium number because:

$$1^1 + 7^2 + 5^3 = 1 + 49 + 125 = 175$$

Task:
Write a Python function that checks whether a given number is a **Disarium Number** or not.

Input:
A positive Integer (n)

Problem Statement – Disarium Number

Output:

- A string “Yes” if the number is a Disarium Number
- A String “No” otherwise

Examples to Check

- 175
- 89
- 12

Problem Statement – Harshad Number

A **Harshad Number (or Niven Number)** is a number that is divisible by the sum of its digits. If a number n can be divided evenly by the sum of its digits, it is considered a Harshad Number.

Task:
Write a Python function that checks whether a given number is a **Harshad Number.**

Input:
A positive Integer (n)

Problem Statement – Harshad Number

Output:

- A string “Yes” if the number is a Harshad Number
- A String “No” otherwise

Examples to Check

- 18
- 21
- 19

Problem Statement – Keith Number

A **Keith Number (or Repfigit Number)** is a number that appears as a term in a special sequence that starts with its digits. The sequence is generated by summing the previous n terms (where n is the number of digits of the number), and the process continues until either the number itself appears in the sequence (in which case it is a Keith Number) or the sum exceeds the number (in which case it is not a Keith Number).

Task:
Write a Python function that checks whether a given number is a Keith Number.

Input:
A positive Integer (n)

Problem Statement – Keith Number

Output:

- A string "Yes" if the number is a Keith Number.
- A string "No" otherwise.

Examples to Check

- 197
- 14

Problem Statement – Longest Substring

Given a string **s**, write a Python function to find the length of the longest substring that contains no repeating characters.

Input:
A string s with minimum length of 1

Output:
An integer representing the length of the longest substring without repeating characters.

Problem Statement – Longest Substring

Examples to Check

- Input: “abcabcbb”
- Output: 3

- Input: “bbbbb”
- Output: 1

- Input: “pwwkew”
- Output: 3

Problem Statement – First Non-Repeating

Write a Python function that finds the first non-repeating character in a given string and returns it. If no such character exists, return None.

Input:
A string s with minimum length of 1

Output:
A single character representing the first non-repeating character, or None if no such character exists.

Problem Statement – First Non-Repeating

Examples to Check

- Input: “swiss”
- Output: “w”

- Input: “hello”
- Output: “h”

- Input: “aabbcc”
- Output: None

Problem Statement – String Compressor

Write a Python function that compresses a string by replacing consecutive occurrences of the same character with the character followed by the count of repetitions. If the compressed version of the string is not smaller than the original, return the original string instead.

Input:
A string s with minimum length of 1

Output:
The compressed string if it is shorter than the original; otherwise, return the original string.

Problem Statement – String Compressor

Examples to Check

- Input: “aabccccccaaa”
- Output: “a2b1c5a3”

- Input: “abc”
- Output: “abc”