Q.1] Given two strings s and t, determine if they are isomorphic.

Two strings s and t are isomorphic if the characters in s can be replaced to get t.

All occurrences of a character must be replaced with another character while preserving the order of characters. No two characters may map to the same character, but a character may map to itself.

**Example 1:**

**Input:** s = "egg", t = "add"

**Output:** true

Solution :

def isomorphic\_strings(s, t):

if len(s) != len(t):

return False

s\_to\_t = {}

t\_to\_s = {}

for i in range(len(s)):

char\_s = s[i]

char\_t = t[i]

if char\_s in s\_to\_t and s\_to\_t[char\_s] != char\_t:

return False

if char\_t in t\_to\_s and t\_to\_s[char\_t] != char\_s:

return False

s\_to\_t[char\_s] = char\_t

t\_to\_s[char\_t] = char\_s

return True

# Example usage

s = "egg"

t = "add"

print(isomorphic\_strings(s, t)) # Output: True

Q.2] Given a string num which represents an integer, return true if num is a ***strobogrammatic number***.

A **strobogrammatic number** is a number that looks the same when rotated 180 degrees (looked at upside down).

**Example 1:**

**Input:** num = "69"

**Output:**

true

Solution :

def is\_strobogrammatic(num):

mapping = {'0': '0', '1': '1', '6': '9', '8': '8', '9': '6'}

left, right = 0, len(num) - 1

while left <= right:

if num[left] not in mapping or mapping[num[left]] != num[right]:

return False

left += 1

right -= 1

return True

# Example usage

num = "69"

print(is\_strobogrammatic(num)) # Output: True

Q.3] Given two non-negative integers, num1 and num2 represented as string, return the sum of num1 and num2 as a string.

You must solve the problem without using any built-in library for handling large integers (such as BigInteger). You must also not convert the inputs to integers directly.

**Example 1:**

**Input:** num1 = "11", num2 = "123"

**Output:**

"134"

Solution :

def addStrings(num1, num2):

result = []

carry = 0

i, j = len(num1) - 1, len(num2) - 1

while i >= 0 or j >= 0 or carry:

digit1 = int(num1[i]) if i >= 0 else 0

digit2 = int(num2[j]) if j >= 0 else 0

carry, digit\_sum = divmod(digit1 + digit2 + carry, 10)

result.append(str(digit\_sum))

i -= 1

j -= 1

result.reverse()

return ''.join(result)

# Example usage

num1 = "11"

num2 = "123"

print(addStrings(num1, num2)) # Output: "134"

Q.4] Given a string s, reverse the order of characters in each word within a sentence while still preserving whitespace and initial word order.

**Example 1:**

**Input:** s = "Let's take LeetCode contest"

**Output:** "s'teL ekat edoCteeL tsetnoc"

Solution :

def reverseWords(s):

words = s.split()

reversed\_words = [word[::-1] for word in words]

reversed\_sentence = ' '.join(reversed\_words)

return reversed\_sentence

# Example usage

s = "Let's take LeetCode contest"

print(reverseWords(s)) # Output: "s'teL ekat edoCteeL tsetnoc"

Q.5] Given a string s and an integer k, reverse the first k characters for every 2k characters counting from the start of the string.

If there are fewer than k characters left, reverse all of them. If there are less than 2k but greater than or equal to k characters, then reverse the first k characters and leave the other as original.

**Example 1:**

**Input:** s = "abcdefg", k = 2

**Output:**

"bacdfeg"

Solution :

def reverseStr(s, k):

char\_list = list(s)

n = len(char\_list)

for i in range(0, n, 2 \* k):

left = i

right = min(i + k - 1, n - 1)

while left < right:

char\_list[left], char\_list[right] = char\_list[right], char\_list[left]

left += 1

right -= 1

return ''.join(char\_list)

# Example usage

s = "abcdefg"

k = 2

print(reverseStr(s, k)) # Output: "bacdfeg"

Q.6 ]

Given two strings s and goal, return true *if and only if* s *can become* goal *after some number of* ***shifts*** *on* s.

A **shift** on s consists of moving the leftmost character of s to the rightmost position.

* For example, if s = "abcde", then it will be "bcdea" after one shift.

**Example 1:**

**Input:** s = "abcde", goal = "cdeab"

**Output:**

true

Solution :

def rotateString(s, goal):

if len(s) != len(goal):

return False

s\_concat = s + s

if goal in s\_concat:

return True

else:

return False

# Example usage

s = "abcde"

goal = "cdeab"

print(rotateString(s, goal)) # Output: True

Q.7] Given two strings s and t, return true *if they are equal when both are typed into empty text editors*. '#' means a backspace character.

Note that after backspacing an empty text, the text will continue empty.

**Example 1:**

**Input:** s = "ab#c", t = "ad#c"

**Output:** true

**Explanation:**

Both s and t become "ac".

Solution :

def process\_string(string):

stack = []

for char in string:

if char == '#':

if stack:

stack.pop()

else:

stack.append(char)

return ''.join(stack)

def backspaceCompare(s, t):

processed\_s = process\_string(s)

processed\_t = process\_string(t)

return processed\_s == processed\_t

# Example usage

s = "ab#c"

t = "ad#c"

print(backspaceCompare(s, t)) # Output: True

Q.8] You are given an array coordinates, coordinates[i] = [x, y], where [x, y] represents the coordinate of a point. Check if these points make a straight line in the XY plane.

**Input:** coordinates = [[1,2],[2,3],[3,4],[4,5],[5,6],[6,7]]

**Output:** true

Solution :

def checkStraightLine(coordinates):

if len(coordinates) <= 2:

return True

x1, y1 = coordinates[0]

x2, y2 = coordinates[1]

initial\_slope = float('inf') if x2 - x1 == 0 else (y2 - y1) / (x2 - x1)

for i in range(2, len(coordinates)):

x, y = coordinates[i]

new\_slope = float('inf') if x - x1 == 0 else (y - y1) / (x - x1)

if new\_slope != initial\_slope:

return False

return True

# Example usage

coordinates = [[1,2],[2,3],[3,4],[4,5],[5,6],[6,7]]

print(checkStraightLine(coordinates)) # Output: True