**Assignment 7**

Q1. 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

Sol.

def isomorphic\_strings(s, t):

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

return False

char\_map\_s = {}

char\_map\_t = {}

for i in range(len(s)):

if s[i] not in char\_map\_s:

char\_map\_s[s[i]] = t[i]

if t[i] not in char\_map\_t:

char\_map\_t[t[i]] = s[i]

if char\_map\_s[s[i]] != t[i] or char\_map\_t[t[i]] != s[i]:

return False

return True

Q2. 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

Sol.

def is\_strobogrammatic(num):

digit\_map = {'0': '0', '1': '1', '6': '9', '8': '8', '9': '6'}

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

while left <= right:

if num[left] not in digit\_map or num[right] not in digit\_map or digit\_map[num[left]] != num[right]:

return False

left += 1

right -= 1

return True

Q3. 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"

Sol.

def addStrings(num1, num2):

p1, p2 = len(num1) - 1, len(num2) - 1

carry = 0

result = ""

while p1 >= 0 or p2 >= 0 or carry != 0:

if p1 >= 0:

carry += int(num1[p1])

p1 -= 1

if p2 >= 0:

carry += int(num2[p2])

p2 -= 1

result += str(carry % 10)

carry //= 10

return result[::-1]

Q4. 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"

Sol.

def reverseWords(s):

words = s.split() # Split the string into a list of words

reversed\_words = [word[::-1] for word in words] # Reverse each word in the list

reversed\_sentence = " ".join(reversed\_words) # Join the reversed words with whitespace

return reversed\_sentence

Q5. 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"

Sol.

def reverseStr(s, k):

result = []

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

group = s[i:i + 2 \* k]

modified\_group = group[:k][::-1] + group[k:]

result.append(modified\_group)

return "".join(result)

Q6. 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

Sol.

def rotateString(s, goal):

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

return False

s\_concatenated = s + s

return goal in s\_concatenated

Q7. 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".

Sol.

def backspaceCompare(s, t):

def process\_string(string):

stack = []

for char in string:

if char != '#':

stack.append(char)

elif stack:

stack.pop()

return "".join(stack)

processed\_s = process\_string(s)

processed\_t = process\_string(t)

return processed\_s == processed\_t

Q8. 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.

Example:

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

**Output:** true

Sol.

def checkStraightLine(coordinates):

if len(coordinates) <= 2:

return True

x1, y1 = coordinates[0]

x2, y2 = coordinates[1]

initial\_slope = (y2 - y1) / (x2 - x1)

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

x, y = coordinates[i]

if (y2 - y) / (x2 - x) != initial\_slope:

return False

return True