**Assignment Questions 7**

**Question 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

**CODE:**

**def** isIsomorphic(s, t):

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

**return** **False**

*# Create two dictionaries to store character mappings*

s\_to\_t **=** {}

t\_to\_s **=** {}

*# Iterate over the characters in both strings*

**for** i **in** range(len(s)):

*# Check if the current characters already have a mapping*

**if** s[i] **in** s\_to\_t **and** s\_to\_t[s[i]] **!=** t[i]:

**return** **False**

**if** t[i] **in** t\_to\_s **and** t\_to\_s[t[i]] **!=** s[i]:

**return** **False**

*# If the characters don't have a mapping, create one*

s\_to\_t[s[i]] **=** t[i]

t\_to\_s[t[i]] **=** s[i]

*# All characters have been checked and no conflicts found*

**return** **True**

**Question 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

**CODE:**

**def** isStrobogrammatic(num):

strobogrammatic\_pairs **=** {'0':'0', '1':'1', '6':'9', '8':'8', '9':'6'}

*# Initialize two pointers at the start and end of the number*

left **=** 0

right **=** len(num) **-** 1

**while** left **<=** right:

*# Check if the pair of digits is a strobogrammatic pair*

**if** num[left] **not** **in** strobogrammatic\_pairs **or** num[right] **!=** strobogrammatic\_pairs[num[left]]:

**return** **False**

*# Move the pointers*

left **+=** 1

right **-=** 1

*# All pairs have been checked and are strobogrammatic*

**return** **True**

**Question 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"

**CODE:**

**def** addStrings(num1, num2):

*# Initialize pointers for the last digit of each number*

i **=** len(num1) **-** 1

j **=** len(num2) **-** 1

carry **=** 0 *# Variable to keep track of the carry*

result **=** [] *# List to store the digits of the sum*

*# Perform digit-wise addition from right to left*

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

*# Get the digits of the two numbers, or 0 if we have reached the beginning of one of the numbers*

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

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

*# Add the digits along with the carry*

digit\_sum **=** digit1 **+** digit2 **+** carry

*# Update the carry if the sum is greater than 9*

carry **=** digit\_sum **//** 10

*# Add the least significant digit to the result list*

result**.**append(str(digit\_sum **%** 10))

*# Move the pointers to the next digits*

i **-=** 1

j **-=** 1

*# Reverse the result list and join the digits to form the final sum as a string*

**return** ''**.**join(result[::**-**1])

**Question 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"

**CODE”**

**def** reverseWords(s):

*# Split the sentence into words*

words **=** s**.**split()

*# Reverse each word in the list*

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

*# Join the reversed words with whitespace*

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

**return** reversed\_sentence

**Question 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"

**CODE:**

**def** reverseStr(s, k):

n **=** len(s)

result **=** list(s) *# Convert string to a list of characters*

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

left **=** i

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

*# Reverse the substring of length k*

**while** left **<** right:

result[left], result[right] **=** result[right], result[left]

left **+=** 1

right **-=** 1

*# Convert the list of characters back to a string*

reversed\_str **=** ''**.**join(result)

**return** reversed\_str

**Question 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

**CODE:**

**def** rotateString(s, goal):

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

**return** **False**

concatenated **=** s **+** s

**if** goal **in** concatenated:

**return** **True**

**else**:

**return** **False**

**Question 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".

**CODE:**

**def** backspaceCompare(s, t):

**def** buildString(string):

stack **=** []

**for** char **in** string:

**if** char **!=** '#':

stack**.**append(char)

**elif** stack:

stack**.**pop()

**return** ''**.**join(stack)

**return** buildString(s) **==** buildString(t)

**Question 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.

**Example 1:**

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

**Output:** true

**CODE:**

**def** checkStraightLine(coordinates):

**if** len(coordinates) **<=** 2:

**return** **True**

x0, y0 **=** coordinates[0]

x1, y1 **=** coordinates[1]

*# Calculate the initial slope*

**if** x1 **-** x0 **!=** 0:

slope **=** (y1 **-** y0) **/** (x1 **-** x0)

**else**:

slope **=** float('inf')

*# Check the slope between each pair of consecutive points*

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

x, y **=** coordinates[i]

**if** x **-** x0 **!=** 0:

current\_slope **=** (y **-** y0) **/** (x **-** x0)

**else**:

current\_slope **=** float('inf')

*# If the current slope is not equal to the initial slope, return False*

**if** current\_slope **!=** slope:

**return** **False**

**return** **True**