**ASSIGNMENT-VII**

**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

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

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

return False

s\_to\_t = {}

t\_to\_s = {}

for c\_s, c\_t in zip(s, t):

if c\_s not in s\_to\_t and c\_t not in t\_to\_s:

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

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

elif s\_to\_t.get(c\_s) != c\_t or t\_to\_s.get(c\_t) != c\_s:

return False

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

Ans: def isStrobogrammatic(num):

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

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

while left <= right:

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

return False

left += 1

right -= 1

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"

Ans: def addStrings(num1, num2):

result = ""

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

carry = 0

while p1 >= 0 or p2 >= 0:

digit1 = int(num1[p1]) if p1 >= 0 else 0

digit2 = int(num2[p2]) if p2 >= 0 else 0

current\_sum = digit1 + digit2 + carry

carry = current\_sum // 10

digit = current\_sum % 10

result = str(digit) + result

p1 -= 1

p2 -= 1

if carry > 0:

result = str(carry) + result

return result

**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"

**Ans:** def reverseWords(s):

words = s.split()

reversed\_words = []

for word in words:

reversed\_word = word[::-1]

reversed\_words.append(reversed\_word)

reversed\_s = " ".join(reversed\_words)

return reversed\_s

**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"

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

s = list(s)

n = len(s)

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

left = i

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

while left < right:

s[left], s[right] = s[right], s[left]

left += 1

right -= 1

return "".join(s)

**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

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

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

return False

s2 = s + s

if goal in s2:

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".

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

s\_stack = []

t\_stack = []

for c in s:

if c != '#':

s\_stack.append(c)

elif s\_stack:

s\_stack.pop()

for c in t:

if c != '#':

t\_stack.append(c)

elif t\_stack:

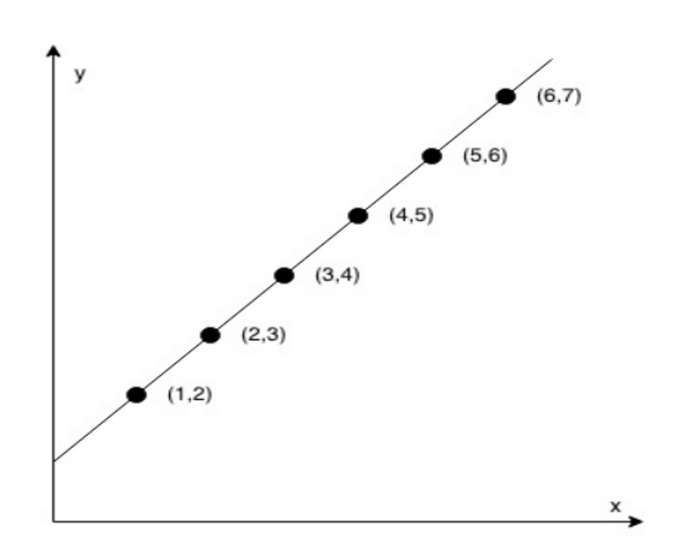
t\_stack.pop()

return s\_stack == t\_stack

**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

Ans: def checkStraightLine(coordinates):

if len(coordinates) < 2:

return True

x = [coord[0] for coord in coordinates]

y = [coord[1] for coord in coordinates]

expected\_slope = (y[1] - y[0]) / (x[1] - x[0])

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

current\_slope = (y[i] - y[i-1]) / (x[i] - x[i-1])

if current\_slope != expected\_slope:

return False

return True