**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

Program: class Solution:

def isIsomorphic(self, s: str, t: str) -> bool:

mapping\_s\_t = {}

mapping\_t\_s = {}

for c1, c2 in zip(s, t):

if (c1 not in mapping\_s\_t) and (c2 not in mapping\_t\_s):

mapping\_s\_t[c1] = c2

mapping\_t\_s[c2] = c1

elif mapping\_s\_t.get(c1) != c2 or mapping\_t\_s.get(c2) != c1:

return False

return True

INPUT: s = "egg", t ="add" OUTPUT : 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

**Program**: **class Solution(object):**

**def isStrobogrammatic(self, num):**

**maps = {("0", "0"), ("1", "1"), ("6", "9"), ("8", "8"), ("9", "6")}**

**i,j = 0, len(num) - 1**

**while i <= j:**

**if (num[i], num[j]) not in maps:**

**return False**

**i += 1**

**j -= 1**

**return True**

**Input:** num = "69"

**Output:** 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"

**Program: class Solution:**

**def addStrings(self, num1: str, num2: str) -> str:**

**ans = []**

**carry = 0**

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

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

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

**if i >= 0:**

**carry += int(num1[i])**

**if j >= 0:**

**carry += int(num2[j])**

**ans.append(str(carry % 10))**

**carry //= 10**

**i -= 1**

**j -= 1**

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

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

**OUTPUT: "134"**

**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"**

**Program: class Solution:**

**def reverseWords(self, s: str) -> str:**

**words = s.split()**

**reversed\_words = []**

**for word in words:**

**reversed\_words.append(word[::-1])**

**return " ".join(reversed\_words)**

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

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

**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"**

**Program: class Solution:**

**def reverseString(self, s: List[str]) -> None:**

**l = len(s)**

**for i in range(l//2):**

**s[i], s[l-1-i] = s[l-1-i], s[i]**

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

**Output: "bacdfeg"**

**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**

**Program: class Solution:**

**def rotateString(self, s: str, goal: str) -> bool:**

**return (len(s) == len(goal) and goal in (s + s) )**

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

**OUTPUT : true**

**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".**

**Program: class Solution:**

**def backspaceCompare(self, S: str, T: str) -> bool:**

**def bckspc(string):**

**stack=[]**

**for i in string:**

**if i =='#':**

**try:**

**stack.pop()**

**except:**

**pass**

**else:**

**stack.append(i)**

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

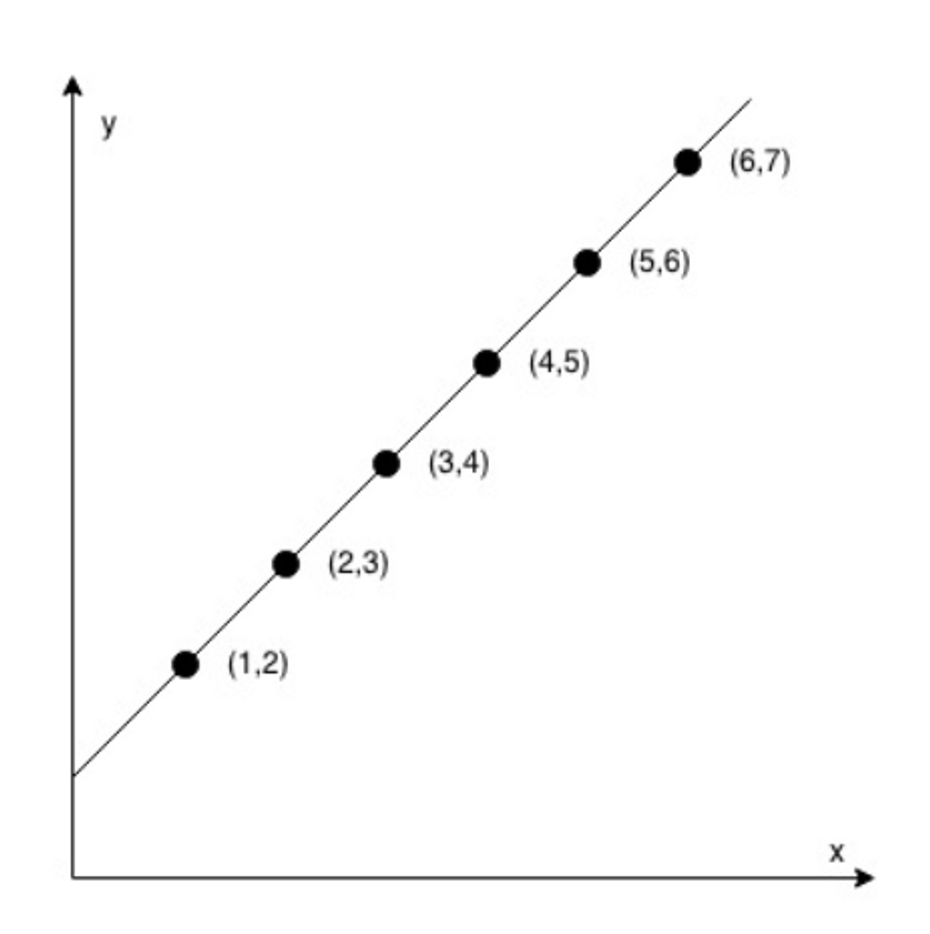
**return (bckspc(S)==bckspc(T))**

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

**Output: true**

**Question 8**

**You are given an array of 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**

**Program:   
class Solution:**

**def checkStraightLine(self, coordinates: List[List[int]]) -> bool:**

**x0, y0, x1, y1 = \*coordinates[0], \*coordinates[1]**

**dx = x1 - x0**

**dy = y1 - y0**

**return all((x - x0) \* dy == (y - y0) \* dx for x, y in coordinates)**

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

**Output: true**