**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

**Solution:**

**Python Code**

def is\_isomorphic(s, t):

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

return False

s\_map = {}

t\_map = {}

for i in range(len(s)):

s\_char = s[i]

t\_char = t[i]

if s\_char not in s\_map:

s\_map[s\_char] = t\_char

elif s\_map[s\_char] != t\_char:

return False

if t\_char not in t\_map:

t\_map[t\_char] = s\_char

elif t\_map[t\_char] != s\_char:

return False

return True

# user input for strings s and t

s = input("Enter the value for s: ")

t = input("Enter the value for t: ")

# function call

print(is\_isomorphic(s, t))

💡 **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

Solution:

Python Code

def is\_strobogrammatic(num):

mapping = {

'0': '0',

'1': '1',

'6': '9',

'8': '8',

'9': '6'

}

left = 0

right = 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

# Get user input for the number num

num = input("Enter a number: ")

# Check if num is strobogrammatic

print(is\_strobogrammatic(num))

**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"

Solution:

Python Code

def addStrings(num1, num2):

result = ""

p1 = len(num1) - 1

p2 = len(num2) - 1

carry = 0

while p1 >= 0 or p2 >= 0:

x1 = int(num1[p1]) if p1 >= 0 else 0

x2 = int(num2[p2]) if p2 >= 0 else 0

temp\_sum = x1 + x2 + carry

result = str(temp\_sum % 10) + result

carry = temp\_sum // 10

p1 -= 1

p2 -= 1

if carry > 0:

result = str(carry) + result

return result

# Get user input for num1 and num2

num1 = input("Enter the first number: ")

num2 = input("Enter the second number: ")

# Add num1 and num2

print(addStrings(num1, num2))

**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"

Solution:

Python Code

def reverseWords(s):

words = s.split()

for i in range(len(words)):

words[i] = words[i][::-1]

return ' '.join(words)

# Get user input for the string s

s = input("Enter a sentence: ")

# Reverse the words in s

print(reverseWords(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"

Solution:

Python Code

def reverseStr(s, k):

s\_list = list(s)

i = 0

while i < len(s):

# Reverse the first k characters

s\_list[i:i+k] = s\_list[i:i+k][::-1]

# Move the pointer to skip the next 2k characters

i += 2 \* k

return ''.join(s\_list)

# Get user input for the string s

s = input("Enter a string: ")

# Get user input for the integer k

k = int(input("Enter an integer k: "))

# Reverse the first k characters for every 2k characters in s

print(reverseStr(s, k))

**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

Solution:

Python Code

def rotateString(s, goal):

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

return False

rotated = s + s

if goal in rotated:

return True

return False

# Get user input for the string s

s = input("Enter the first string: ")

# Get user input for the string goal

goal = input("Enter the second string: ")

# Check if s can be transformed into goal after some number of shifts

print(rotateString(s, goal))

**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".

Solution:

Python Code

def buildString(s):

stack = []

for c in s:

if c != '#':

stack.append(c)

elif stack:

stack.pop()

return ''.join(stack)

def backspaceCompare(s, t):

return buildString(s) == buildString(t)

# Get user input for the string s

s = input("Enter the first string: ")

# Get user input for the string t

t = input("Enter the second string: ")

# Compare the strings after considering backspaces

print(backspaceCompare(s, 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

A picture containing line, screenshot, diagram, plot

Description automatically generated

Solution:

Python Code

def checkStraightLine(coordinates):

if len(coordinates) < 2:

return True

x1, y1 = coordinates[0]

x2, y2 = coordinates[1]

# Calculate the initial slope

if x2 - x1 != 0:

slope = (y2 - y1) / (x2 - x1)

else:

slope = float('inf')

# Check the slope for the remaining points

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

x, y = coordinates[i]

# Calculate the slope between the current point and the first point

if x - x1 != 0:

curr\_slope = (y - y1) / (x - x1)

else:

curr\_slope = float('inf')

# If the slope is different, return False

if curr\_slope != slope:

return False

return True

# Get user input for the coordinates array

coordinates = []

num\_points = int(input("Enter the number of points: "))

for \_ in range(num\_points):

x = int(input("Enter x-coordinate: "))

y = int(input("Enter y-coordinate: "))

coordinates.append([x, y])

# Check if the points form a straight line

print(checkStraightLine(coordinates))