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

Soln:

```
def isIsomorphic(s, t):
    if len(s) != len(t):
        return False

    s_to_t = {}
    t_to_s = {}

    for char_s, char_t in zip(s, t):
        if (char_s in s_to_t and s_to_t[char_s] != char_t) or (char_t in t_to_s and t_to_s[char_t] !=
char_s):
            return False
        s_to_t[char_s] = char_t
        t_to_s[char_t] = char_s

    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

Soln:

```
def isStrobogrammatic(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
```

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"

Soln:

```
def addStrings(num1, num2):

    p1 = len(num1) - 1

    p2 = len(num2) - 1

    carry = 0
```

```
    result = ""

    while p1 >= 0 or p2 >= 0:
        digit1 = int(num1[p1]) if p1 >= 0 else 0
        digit2 = int(num2[p2]) if p2 >= 0 else 0
        digit_sum = digit1 + digit2 + carry
        carry = digit_sum // 10
        result = str(digit_sum % 10) + result
        p1 -= 1
        p2 -= 1

    if carry:
        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"

Soln:

```
def reverseWords(s):
    words = s.split()
    reversed_words = [word[::-1] for word in words]
    reversed_sentence = ' '.join(reversed_words)  # Join the reversed words back together
    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"

Soln:

```
def reverse_string(s, k):
    # Convert the string into a list since strings are immutable in Python
    s = list(s)

    # Iterate over the string in steps of size 2k
    for i in range(0, len(s), 2 * k):
        # Reverse the first k characters if there are at least k characters remaining
```

```
    if i + k <= len(s):
        s[i:i+k] = reversed(s[i:i+k])

# Convert the list back to a string and return the result
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

Soln:

```
def can_shift(s, goal):
    # Checking if goal is a substring of s + s
    if goal in s + s:
        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".

Soln:

```
def process_string(s):
    stack = []
    for char in s:
        if char != '#':
            stack.append(char)
        elif stack:
            stack.pop()
    return ''.join(stack)


def backspace_compare(s, t):
    return process_string(s) == process_string(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.

Soln:

```
def check_straight_line(coordinates):
    if len(coordinates) <= 2:
        return True

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

    # Calculate the slope between the first two points
    if x1 - x0 == 0:
        slope = float('inf')
    else:
        slope = (y1 - y0) / (x1 - x0)

    # Check the slope between subsequent points
    for i in range(2, len(coordinates)):
        x, y = coordinates[i]

        if x1 - x == 0:
            current_slope = float('inf')
        else:
            current_slope = (y1 - y) / (x1 - x)
```

```
    # If the slope is not the same, the points do not form a straight line
    if current_slope != slope:
        return False

return True
```