# ASSIGNMENT 7 (STRINGS)

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.

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

    char_map = {}
    used_chars = set()

    for i in range(len(s)):
        char_s = s[i]
        char_t = t[i]

        if char_s in char_map:
            if char_map[char_s] != char_t:
                return False
        else:
            if char_t in used_chars:
                return False

            char_map[char_s] = char_t
            used_chars.add(char_t)

    return True
```

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).

```
def is_strobogrammatic(num):
    strobogrammatic_pairs = {'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_pairs or num[right] not in strobogrammatic_pairs:
            return False
        if num[left] != strobogrammatic_pairs[num[right]]:
            return False
        left += 1
        right -= 1

    return True
```

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.

```
def add_strings(num1, num2):
    result = []
    carry = 0
    i, j = len(num1) - 1, len(num2) - 1

    while i >= 0 or j >= 0 or carry:
        digit1 = int(num1[i]) if i >= 0 else 0
        digit2 = int(num2[j]) if j >= 0 else 0

        # Perform digit-by-digit addition
        total = digit1 + digit2 + carry
        carry = total // 10
        digit = total % 10
        result.append(str(digit))

        i -= 1
        j -= 1
    return ''.join(result[::-1])
```

4. Given a string s, reverse the order of characters in each word within a sentence while still preserving whitespace and initial word order.

```
def reverse_words(s):
    words = s.split()  # Split the sentence into words
    reversed_words = [word[::-1] for word in words]  # Reverse each word
    reversed_sentence = ' '.join(reversed_words)  # Join the reversed words with whitespace
    return reversed_sentence
```

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.

```
def reverse_string(s, k):
    chars = list(s)  # Convert the string to a list of characters
    n = len(chars)

    for i in range(0, n, 2 * k):
        left = i
        right = min(i + k - 1, n - 1)

        while left < right:
            chars[left], chars[right] = chars[right], chars[left]
            left += 1
            right -= 1

    return ''.join(chars)
```

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.

```
def can_shift(s, goal):
    if len(s) != len(goal):
        return False

    shifted_s = s + s
    return goal in shifted_s
```

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.

```
def backspace_compare(s, t):
    def process_string(string):
        processed = []
        for char in string:
            if char != '#':
                processed.append(char)
            elif processed:
                processed.pop()
        return ''.join(processed)

    return process_string(s) == process_string(t)
```

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.

```
def check_straight_line(coordinates):
    n = len(coordinates)

    # Calculate the slope between the first two points
    x1, y1 = coordinates[0]
    x2, y2 = coordinates[1]
    slope = (y2 - y1) / (x2 - x1) if x2 - x1 != 0 else float('inf')

    # Check the slopes between subsequent points
    for i in range(2, n):
        x1, y1 = coordinates[i-1]
        x2, y2 = coordinates[i]
        current_slope = (y2 - y1) / (x2 - x1) if x2 - x1 != 0 else float('inf')

        # If the current slope is not equal to the initial slope, return False
        if current_slope != slope:
            return False

    return True
```