National University of Singapore
School of Computing
TT1003: Programming Methodology Clinic
Semester II, 2024/2025

Optional Practices : Iteration and Recursion

Background

This is an **optional** PE1-style practice question created by Zhu Ming.

Problem 1: Sorting Two Numeric Strings (Ideal time: 5 minutes)

Write a function, `sort_two_numeric_strings` that takes two strings of numerical digits, each already arranged in ascending order, as input. The function should return a string containing all digits from both input strings, merged while maintaining sorted order.

For an iterative approach, the entire solution must be implemented purely iteratively. For a recursive approach, the entire solution must be implemented purely recursively.

```
>>> sort_two_numeric_strings('123', '456')
'123456'
>>> sort_two_numeric_strings('111', '456')
'111456'
>>> sort_two_numeric_strings('123', '')
'123'
```

```
def sort_two_numeric_strings(s1, s2):
    if s1 == '' or s2 == '':
        return s1 + s2
    if s1[0] < s2[0]:
        s1, s2 = s2, s1
    return s2[0] + sort_two_numeric_strings(s1, s2[1:])

def sort_two_numeric_strings_iter(s1, s2):
    res = ''
    while s1 and s2:
        if s1[0] >= s2[0]:
            s2, s1 = s1, s2
        res += s1[0]
        s1 = s1[1:]
    return res + s1 + s2
```

Problem 2: Sorting Two Number (Ideal time: 10 minutes)

Write a function, `sort_two_num` that takes two **positive** integers, each with digits arranged in ascending order, as input. The function should return an integer containing all the

digits from both input integers, merged while maintaining sorted order.

For an iterative approach, the entire solution must be implemented purely iteratively. For a recursive approach, the entire solution must be implemented purely recursively. Input is guaranteed to be positive integers, $n > 0$.

```
>>> sort_two_num(123, 456)
123456
>>> sort_two_num(111, 456)
111456
```

```
def sort_two_num(n1, n2):
    if n1 == 0 or n2 == 0:
        return n1 + n2
    if n1 % 10 < n2 % 10:
        n1, n2 = n2, n1
    return n2 % 10 + 10 * sort_two_num(n1, n2 // 10)

def sort_two_num_iter(n1, n2):
    res = 0
    place = 1
    while n1 and n2:
        if n1 % 10 >= n2 % 10:
            n2, n1 = n1, n2
        res += place * (n1 % 10)
        n1 //= 10
        place *= 10
    return res + place * n1 + place * n2
```

Problem 3: Palindrome (Ideal time: 5 minutes)

Write a function, `min_char_required` that takes in a string and determine the minimum number of characters that must be added to the string to make it a palindrome. A palindrome is a string that reads the same forwards and backwards.

For a recursive approach, the entire solution must be implemented purely recursively. (Iteration is too hard for this question)

```
def min_char_required(s):
    if s == s[::-1]:
        return 0
    elif s[0] == s[-1]:
        return min_char_required(s[1:-1])
    else:
        left = 1 + min_char_required(s[1:])
        right = 1 + min_char_required(s[:-1])
        return min(left, right)
```