INTRODUCTION TO DATA STRUCTURES

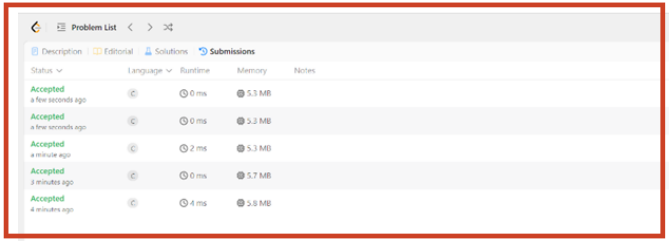
Leetcode: 20. Valid Parentheses & 66. Plus One in C

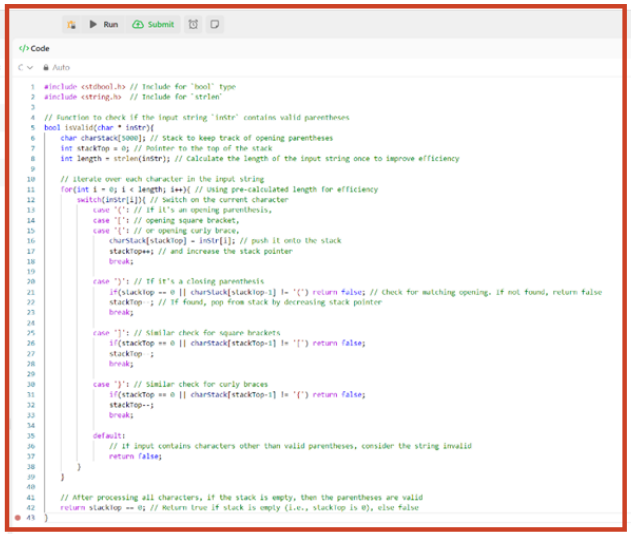
HW0_D84099084_賴澍雨

Shih Yu Lai

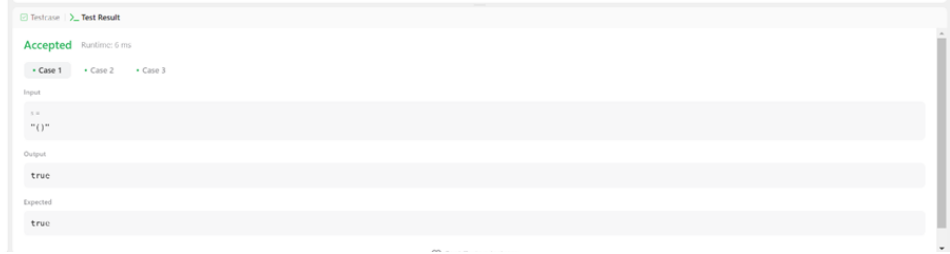
20. Valid Parentheses

提交畫面截圖





程式碼與註解



20. Valid Parentheses

程式碼概念說明

1. 使用的資料結構

- 字符栈（Stack）：字符栈是這個程式的核心資料結構，用於臨時存儲遇到的開括號（、[、{，直到相對應的閉括號)、]、}被找到。
- 棧是一種後進先出（LIFO）的資料結構，最後加入的元素將是第一個被移除。

2.程式邏輯與演算法

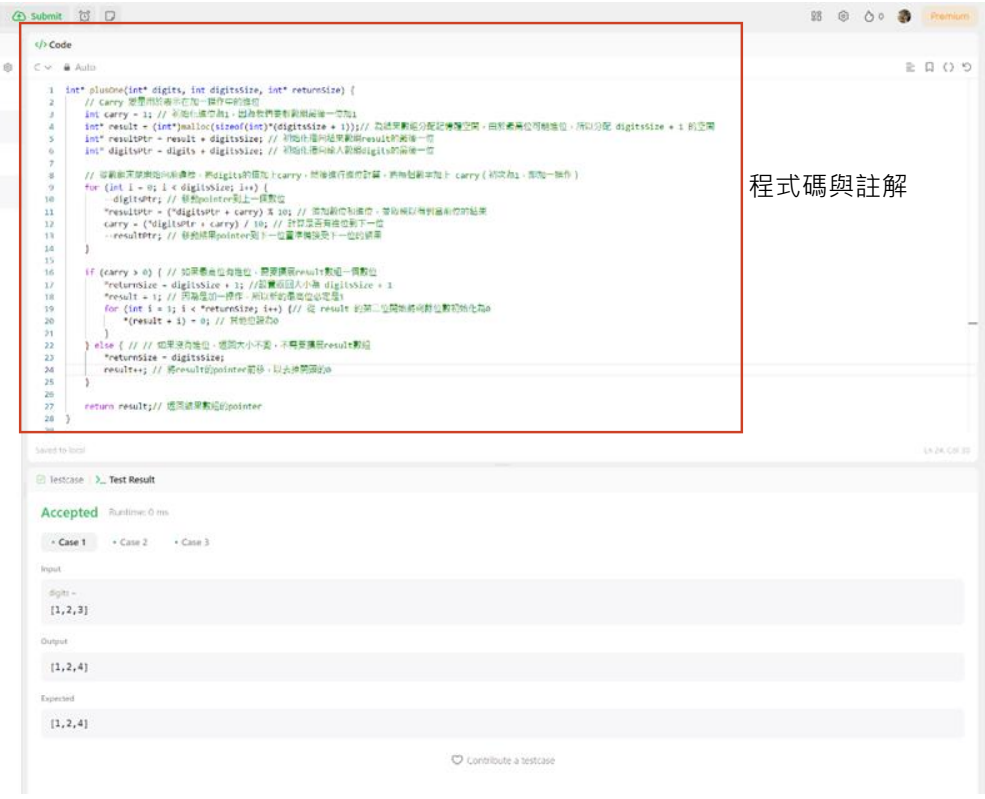
- 程式先計算輸入字符串的長度，以避免在每次迴圈中重複計算。
- 使用 for 迴圈和 pointer ‘inStrPtr ‘逐字符遍歷輸入字符串。
- 對於每個字符，使用 switch 語句判斷其類型：
  - 如果是開括號（(、[、{），則將其推入棧中，並將棧頂 pointer ‘stackPtr ‘向前移動。
  - 如果是閉括號（)、]、}），則檢查棧頂元素是否為對應的開括號。如果是，則將棧頂元素彈出（即將 stackPtr 向後移動）。
  - 如果不是或者棧為空（stackPtr == charStack），則返回 false，表示括號不匹配。
- 循環結束後，如棧不為空（即stackPtr不指向初始位置 charStack），則有未匹配開括號，返回 false。否則所有括號正確配對且棧為空，返回 true。

3.pointer運作

- pointer ‘stackPtr ‘：這是一個指向棧頂的 pointer，用於添加或移除棧中的元素。當有新的開括號被推入棧時， stackPtr 向前移動；當找到匹配的閉括號時， stackPtr 向後移動，以彈出開括號。
- pointer ‘inStrPtr ‘：這是一個指向當前正在處理的輸入字符串字符的 pointer。使用它來遍歷整個字符串，而無需修改原始字符串 inStr 的內容。

66. Plus One

提交畫面截圖



程式碼與註解

66. Plus One

程式碼概念說明

1.使用的資料結構

- **整數(int) pointer**：digits 和 result 都是指向整數的 pointer，分別表示輸入的數字和計算結果。
- **動態分配array**：通過 malloc 函數分配一個動態 array來存儲結果，大小為 digitsSize + 1 以應對可能的進位情況。

2.程式邏輯與演算法

- **進位處理**：通過一個名為 carry 的變量來跟蹤是否需要進位。初始設為 1，因為我們要給最後一位數字加一。
- **反向遍歷**：從 array最後一位開始遍歷，使用 pointer操作來更新數位值並處理進位。
- **結果擴展**：如果在遍歷完成後仍有進位（即array的所有數位都是9），則需要將結果 array擴展一位，新的最高位設為 1，其他位設為 0。
- **返回值設置**：通過 returnSize pointer參數返回結果 array的大小。

3.pointer運作

- **pointer算術**：通過增加或減少 pointer值來移動 pointer，這使我們能夠在不修改原始 array的情況下遍歷 array。
- **賦值與更新**：使用解引用操作符 * 來訪問和更新 pointer指向的值。
- **結果pointer前移**：如果沒有進位，結果 array的實際大小與輸入 array相同，因此將 result pointer前移，以指向正確的開始位置。