Home work #1

1. Delete Strictly Increasing Sequences

用於判斷是否為數字字元('0'~'9')

用於輸出數字，因printf較慢，故以此替代

讀取題目的條件N與T(M)，使用getchar()讀，因scanf過慢，讀到非數字就停止



配置記憶體與主要運作的for迴圈



讀取輸入數列中的數字。



核心程式碼

if (i < M - 1)代表前面的數字不夠下面的for迴圈去檢查，所以直接continue;進行下一個數字的回合，數字足夠，就會跑

for (j = 1; j < M && arr[i - j + 1] > arr[i - j];++j);若在中途被arr[i - j + 1] > arr[i - j]給break掉，就代表從arr的index為i到i-t+1之間並非遞增數列，反之，如下圖則為遞增數列

則會跑完整個for迴圈，導致j == M，接著就要刪除此遞增數列，但不須實際刪除記憶體空間，只需把i(index)指到

i-M處，下一次讀入數字就被覆蓋掉了，因為刪除了數列，所以數列總數N也要減去M



輸出結果



1. Calculator

一些實用的macro，讓程式碼更簡潔，用法如同之後程式碼。



儲存輸入字串與轉後序使用的stack



預先處理動作



output將轉後序的結果存入output內的stack，因為之後要算出結果，所以不能直接印出，my\_strlen(input);回傳string長度，內部實作:



判斷是否invalid，我使用2個function去check，分別為檢查括號匹配的bool check\_brackets(char\* input,int length)與檢查運算子左右兩邊是否有運算元的bool check\_operator(char\* input,int length)以下是內部實作:





以上2個函式都回傳true後，就開始演算法的步驟，先放入'('到stack內，並在input最後加上')'，因為加入新字元，故++len，'\0'推到下一格



主要for迴圈，跑完整個input即可



數字處理

output重載<<，這裡先將class內的output陣列當作stack用，與之後計算數字用的stack不同。



回到主迴圈，接下來是演算法的核心



priority function的內部實作:



主迴圈結束後，已成功完成後序轉換，呼叫output.print\_result();即可。

內部實作:





1. 完整程式碼連結(gist)

[Delete Strictly Increasing Sequences](https://gist.github.com/frakw/e2c2edd7169f6cd44ae6b7666dcf2bec)

[Calculator](https://gist.github.com/frakw/db7a5e08e4a52e70e962fb27f09444f9)

[imgur圖片](https://imgur.com/a/OfwLOQ2)





