當函式 p 被呼叫的時候，傳入 i 跟 N 兩個初始值
情況一:i<N
進到 return 程式由左邊執行到右邊
先得到 i<N 成立 來到 printf 會把 i 印出來且 printf 回傳值等於輸出的字元數大於 0 成立 接著到 !p(i+1,N)，會呼叫下一個 p 函式。
進到下一個 p 函式後又再次檢查 i<N 成立，接著 printf，但這次 printf 的結果是原本的 i+1，printf 後又呼叫了下一個 p 函式，以此類推，直到呼叫到 i==N 時 ，不符合 i<N，對於 and 運算符號來說不管後面的結果如何那結果都會是 0，因此就不做後面的 printf 跟呼叫下一個 p 了。
也就是說這一段程式碼會 printf i,i+1,i+2 直到最後一個數值 i=N-1 被 printf 出來。
接著到了 or 運算符後面，從剛剛結束的 i==N 開始印出，印出 i=N 後，因為 printf 的輸出字符數>0->回傳值>0 整個 p 的回傳值為 1，!p(i+1,N) 也就是 0。
後面碰到 or 運算符，因為前面是 0，所以需要判斷後面，
也就是說會 printf i=N-1，重複上述步驟一路往回推把之前沒做完的 p 函式做完，
也就是說這一段 code 會從 i==N 開始 printf 到最後一個 i==i. 最後的 return 值會因為 or 運算符後的 printf 輸出字元數>1->回傳值>1，所以最後的回傳值在 or 運算下等於 1。
並在最後的畫面呈現出從 i 一路印到 N-1 再從 N 一路印到 i 的畫面
情況 2:i>=N:
由於 i 不會小於 N,所以前面 and 運算符的 printf 不會運作,只會做一次 or 運算符後面得 printf,也就是只會印出 i 本身。