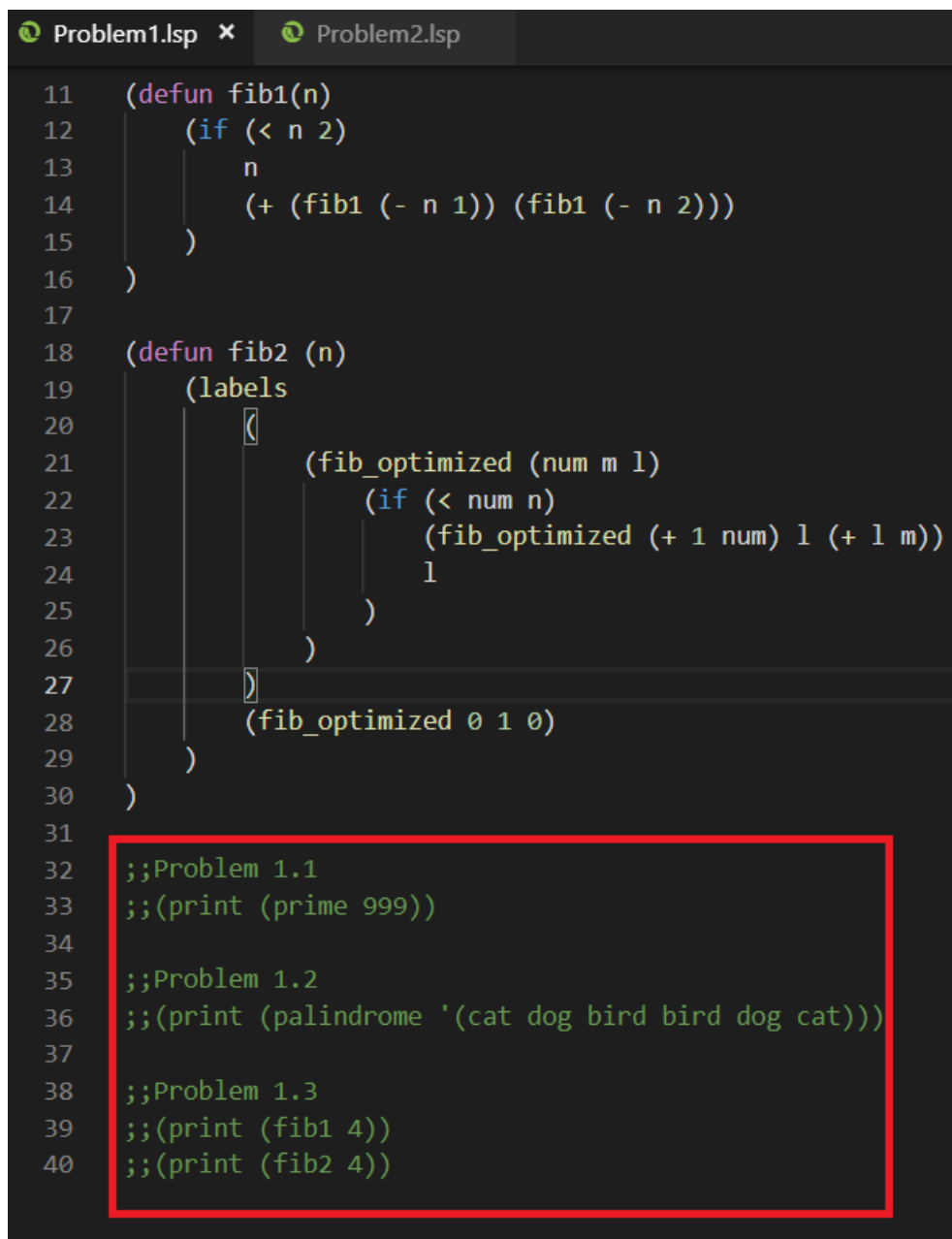


# 程式語言 HW1 報告

陳俊宇 F64051114 資訊 109

## ● 執行環境步驟

這邊先說明一下我的作業寫法，我是一個 Problem 用一個.lsp 檔寫，Problem1.1、1.2、1.3 都實作在 Problem1.lsp 中。Problem1.lsp 的 print function 都先註解起來，如要檢測某個 function 請打開註解，如下圖紅框所示。



```
11 (defun fib1(n)
12   (if (< n 2)
13     n
14     (+ (fib1 (- n 1)) (fib1 (- n 2)))
15   )
16 )
17
18 (defun fib2 (n)
19   (labels
20     ((fib_optimized (num m l)
21      (if (< num n)
22          (fib_optimized (+ 1 num) l (+ 1 m))
23          1
24        )
25     )
26   )
27   (fib_optimized 0 1 0)
28 )
29 )
30
31
32 ;;Problem 1.1
33 ;;(print (prime 999))
34
35 ;;Problem 1.2
36 ;;(print (palindrome '(cat dog bird bird dog cat)))
37
38 ;;Problem 1.3
39 ;;(print (fib1 4))
40 ;;(print (fib2 4))
```

我是在 windows 實作。壓縮檔內包含 Problem1.lsp 與 Problem2.lsp，請將這兩個檔案解壓縮至桌面，並開啟 windows 命令提示字元，按照下

圖方式輸入可執行 Problem1.lsp (Problem1.lsp 放於桌面, Problem2.lsp 以此類推)。

```
C:\Users\user>cd c:\Documents and Settings\user\Desktop
c:\Documents and Settings\user\Desktop>sbcl --script Problem1.lsp
```

如要 trace 某檔案的某 function(此以 trace Problem1.lsp 的 fib1 function 為例, Problem1.lsp 放於桌面), 請依下圖方式輸入即可。

```
C:\Users\user>cd c:\Documents and Settings\user\Desktop
c:\Documents and Settings\user\Desktop>sbcl
This is SBCL 1.1.4.0.mswin.1288-90ab477, an implementation of ANSI Common Lisp.
More information about SBCL is available at <http://www.sbcl.org/>.

SBCL is free software, provided as is, with absolutely no warranty.
It is mostly in the public domain; some portions are provided under
BSD-style licenses. See the CREDITS and COPYING files in the
distribution for more information.

This is experimental prerelease support for the Windows platform: use
at your own risk. "Your Kitten of Death awaits!"
* (load "Problem1.lsp")
T
* (trace fib1)
(FIB1)
* (fib1 3)
0: (FIB1 3)
1: (FIB1 2)
2: (FIB1 1)
2: FIB1 returned 1
2: (FIB1 0)
2: FIB1 returned 0
1: FIB1 returned 1
1: (FIB1 1)
1: FIB1 returned 1
0: FIB1 returned 2
2
*
```

## ● 程式碼說明

### ■ Problem 1.1

判斷 prime 我使用遞迴實作, 我的方法是從要檢驗的數(n)減 1 開始遞迴檢查是否為 n 的因數, 在效率上不是太好, 若從 2 開始往上檢查會更好, 因為較小的數有較大的機率當因數。透過 or 與 and 邏輯判斷, 讓程式碼看起來極為精簡。Trace 結果如下圖。(以 prime 4 為例)

```
* (prime 4)
0: (PRIME 4)
1: (PRIME 4 2)
1: PRIME returned NIL
0: PRIME returned NIL
NIL
*
```

可以看出當找到 2 是 4 的因數時，開始回傳 NIL 回上一層 function call，接著找出答案。

### ■ Problem 1.2

判斷回文，我是透過 LISP 內建 reverse function，若 reverse 後還是相等則回傳 T(true)，否則回傳 NIL(false)。

### ■ Problem 1.3

fib1 我採用上課投影片的 function 實作，是傳統遞迴方式的呈現，迴圈終止條件為  $n < 2$ ，若  $n < 2$  則回傳 n，trace 結果如下圖。(以 fib1 4 為例)

```
* (fib1 4)
0: (FIB1 4)
1: (FIB1 3)
2: (FIB1 2)
3: (FIB1 1)
3: FIB1 returned 1
3: (FIB1 0)
3: FIB1 returned 0
2: FIB1 returned 1
2: (FIB1 1)
2: FIB1 returned 1
1: FIB1 returned 2
1: (FIB1 2)
2: (FIB1 1)
2: FIB1 returned 1
2: (FIB1 0)
2: FIB1 returned 0
1: FIB1 returned 1
0: FIB1 returned 3
3
```

Fib2 的運作方式，我以下表呈現。

num	m	l
0	1	0
1	0	1
2	1	1
3	1	2
4	2	3
5	3	5

變數 l(最右邊那行)，即為費式數列，透過上表紅色箭頭可清楚看出此遞迴運作方式。遞迴終止條件為  $num < n$ ，終止條件觸發則回傳變數 l 的值(也就是我們要的答案)。tail recursion 在我的理解，就是不在該層要做的運算還未做完時就 call 下一個 function，以此方式將不造成某個 function call 做完需回去將尚未計算完的上一層 function call 做完。trace 結果如下圖。(以 fib2 4 為例)

```
* (fib2 4)
0: (FIB2 4)
0: FIB2 returned 3
3
```

## ■ Problem 2

剛開始先用 `if` 判斷遞迴終止條件，若為空或是單一數字則直接 `return`，若非空或非單一數字，則透過 `truncate function`，將 `number sequence` 切開，以此遞迴實作。比較方便的是 LISP 有 `merge function` 可直接使用，第一個參數傳入 `number sequence` 的 `type`，第二與第三個參數傳入兩條要 `merge` 的 `number sequences`，而經過一再對切，到最後切到剩單一元素，再 `merge` 成由小到大排列的 `number sequence`，若想改成由大到小排序，則只需將 `Problem2.lsp` 的第 17 行改成 `#>` 即可。