Programming Language HW1 – LISP

資訊系三年級 邵鏡軒 F74051035

Problem 1: prime number

0 (defun prime (n)

1 (if (= 1 n)

2 (progn

3 (format t "~a is not a prime number~%" n)

4 (return-from prime)

5 )

6 )

7 (if (= 2 n)

8 (progn

9 (format t "~a is a prime number~%" n)

10 (return-from prime)

11 )

12 )

13 (if (= 0 (mod n 2))

14 (format t "~a is not a prime number~%" n)

15 (do ((i 3 (+ i 2)))

16 ((>= i n) (format t "~a is a prime number~%" n))

17 (if (= 0 (mod n i))

18 (progn

19 (format t "~a is not a prime number~%" n)

20 (return-from prime)

21 )

22 )

23 )

24 )

25 )

從3開始每次+2檢查可不可以被除盡

**Program 執行方式**: 測資已經寫在.lsp檔裡

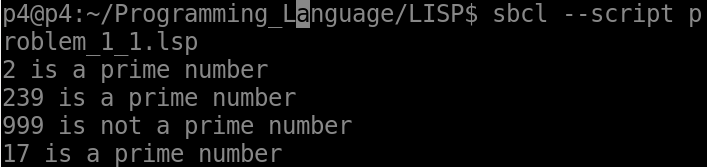
$ sbcl --script problem\_1\_1.lsp

2 is a prime number

239 is a prime number

999 is not a prime number

17 is a prime number



Problem 1.2: palindrome

0 (defun palindrome (input)

1 (if (equal input (reverse input))

2 (format t "~a is a palindrome~%" input)

3 (format t "~a is not a palindrome~%" input)

4 )

5 ; or you can use dolist with (append)

6 )

7

8 (palindrome '(a b c ) )

9 (palindrome '(m a d a m))

10 (palindrome '(cat dog))

11 (palindrome '())

12 (palindrome '(cat dog bird bird dog cat))

迴文檢查方式: 先reverse再用equal看原本的list跟reverse後的list一不一樣，如果一樣那就是迴文

**Program執行方式**: 測資已經寫在.lsp裡

$ sbcl --script problem\_1\_2.lsp

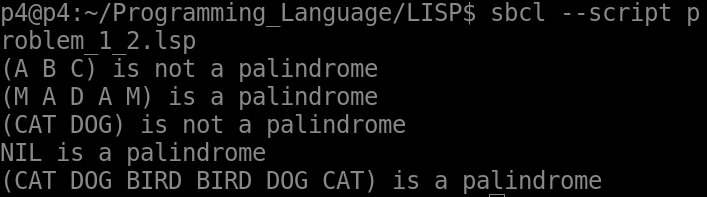
(A B C) is not a palindrome

(M A D A M) is a palindrome

(CAT DOG) is not a palindrome

NIL is a palindrome

(CAT DOG BIRD BIRD DOG CAT) is a palindrome



Problem 1.3

Fib1:

0 (defun fib1 (n)

1 (let ((a 0)(b 0))

2 (if (= n 0)

3 0

4 (if (<= n 2)

5 1

6 (progn

7 (setf a (fib1 (- n 1)))

8 (setf b (fib1 (- n 2)))

9 (+ a b)

10 )

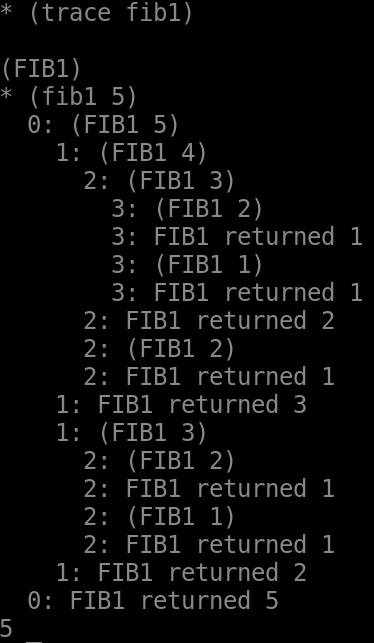
11 )

12 )

13 )

14 )

把n拆解成n-1跟n-2，再把兩個的返回值相加，然後回傳



用trace可以看出一層一層的往下呼叫

**Program 執行方式**: 這次進到sbcl去操作

$ sbcl

\* (貼上上面的程式碼)

\* (fib1 1)

1

\* (fib1 5)

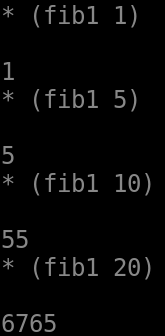
5

\* (fib1 10)

55

\* (exit)

$



Fib2:

0 (defun tail\_fib2 (now n a b)

1 (if (= now n)

2 (format t "~a~%" b)

3 (tail\_fib2 (+ now 1) n b (+ a b))

4 )

5 )

6

7 (defun fib2 (n)

8 (if (= n 0)

9 (format t "0~%")

10 (tail\_fib2 1 n 0 1)

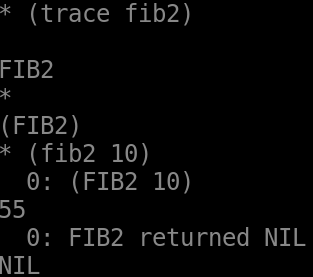
11 )

12 )

1. 由小算到大，now是現在算到的數，n是目標數，a,b是兩個相鄰的數字，每次會更改。

2. 某次函數的結束時都是call自己函數，並且不會再用到任何在這個stack的資料，因此達到tail recursion

3. 由fib2去call sub function



透過trace可看出沒有加深層數的遞迴呼叫

**Program 執行方式**: 這次進到sbcl去操作

$ sbcl

\* (貼上上面的程式碼)

\* (fib2 1)

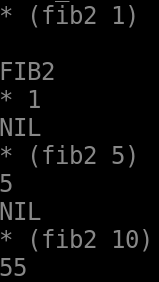
1

\* (fib2 5)

5

\* (fib2 10)

55



Problem 2

0 (defun mergesort (n numbers)

1 (if (<= n 1)

2 (return-from mergesort numbers)

3 )

4 (let ((leng 0)(leng2 0)(list1 nil)(list2 nil))

5 (setf leng (floor n 2))

6 (setf leng2 (+ (floor n 2)(mod n 2)))

7 (setf list1 (mergesort leng (nthcdr leng2 (reverse numbers))))

8 (setf list2 (mergesort leng2 (nthcdr leng numbers)))

9 (setf numbers nil)

10 (do ()

11 ((or (= leng 0)(= leng2 0))

12 (if (= leng 0)

13 (dolist (x list2)

14 (setf numbers (cons x numbers))

15 )

16 (dolist (x list1)

17 (setf numbers (cons x numbers))

18 )

19 )

20 (setf numbers (reverse numbers))

21 )

22 (if (<= (car list1)(car list2))

23 (progn

24 (setf numbers (cons (car list1) numbers ))

25 (setf leng (- leng 1))

26 (setf list1 (cdr list1))

27 )

28 (prong

29 (setf numbers (cons (car list2) numbers ))

30 (setf leng2 (- leng2 1))

31 (setf list2 (cdr list2))

32 )

33 )

34 )

35 )

36 (return-from mergesort numbers)

37 )

38

39; main function

40 (let ((n(read))(numbers))

41 (setf numbers

42 (do ((i 0 (+ i 1))(tmp nil))

43 ((>= i n)(reverse tmp))

44 (setf tmp (cons (read) tmp))

45 )

46 )

47 (format t "~{~A ~}~%" (mergesort n numbers))

48 )

基本上使用跟一般的mergesort一樣的做法，不斷的切開排列再合併，而在lisp用的比較特別的做法有兩個:

(1)把一個list切成一半時，第一半是用nthcdr，另一半是先reverse再nthcdr，如此一來就可以分成前後兩半

(2)每次加入新的element到list時是加在最前面(先從大排到小)，然後最後再reverse回來，如此一來就變回從小排到大

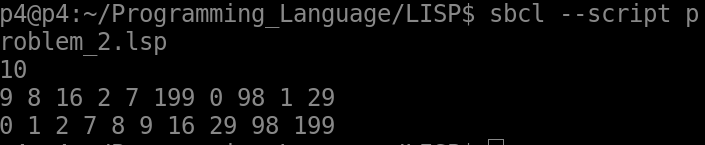
其中幾個比較特別的函式是nthcdr, cons(不是用list或append)

**Program執行方式**: 先輸入要n個數字，再輸入n個數字

$ sbcl --script problem\_2.isp

10

9 8 16 2 7 199 0 98 1 29



Problem 3

(程式碼太長就不在這裡貼出)

**匹配:**

先把檔案A的每一行去跟B的每一行比對，得到A的每一行分別跟B的哪幾行配對到，比如A的第3行跟B的1 3 6 7 13行配對到，如此得到一個list (1 3 6 7 13)，然後可以得到n個list (n為A的行數)，然後從這n個list中各選出0個或1個element，排成最長的由小到大的序列。比如從第1個list取3，第2個list取5，第5個list取8->得到(3 5 8)以及來源。

**印出:**

印出的方式為用B的(3 5 8)行去找到跟A匹配的行數，找到之前把A的都以-“string”印出，再把B的都匹配前的行數以+”string”印出，然後印出匹配的那行”string”。接著再印A，再印B，再印匹配…最後把B的第8行印出來後，剩下的A都以-“string”印出，以及剩下的B以+”string”印出

**Program執行方式**: 如果要修改輸入檔名，請在程式碼的第一二行做修改(預設為file1.txt 以及 file2.txt)

$ sbcl --script problem\_3.lsp

-#include <stdio.h>

+#include <iostream>

+using namespace std;

int main() {

- printf(“Hello World”);

+ cout << “Hello World” << end;

return 0;

}

