Programming Language HW1 LISP 報告

資訊109 F74054067 柳孟芸

1. 執行環境：OS X
   Command Line Interface : 終端機

2. 程式碼解說：

◎Problem 1.1

```
;;; file: problem1_1.lsp

(defun forall(list func)
    (if (null list)
        T
        (and (funcall func (car list))
            (forall (cdr list) func) )
    )
)

(defun nums(start stop)
    (setq L nil)
    (loop ( when(> start stop) (return (reverse L)) )
        (setq L (cons start L) )
        (incf start)    ;+1
    )
)

(defun prime_or_not(n)
    (and
    (> n 1)  ;picky case so we don't say numbers <=1 are prime
    ;;A number is prime if it is not divisible by any number in the range 2..(floor (sqrt n))
    (forall (nums 2 (floor (sqrt n)))
        #'(lambda (divisor) (not (= (mod n divisor) 0)))
    )
    )
)

(defun prime(n)
    (if (prime_or_not n)
        (format t "~D is a prime number.~%" n)
        (format t "~D is not a prime number.~%" n)
    )
)

(prime 2)(prime 239)(prime 999)(prime 17)
```

A boolean function forall( ) :
which takes a list and a predicate and returns true if and only if the predicate returns true for every element in the list.

The function nums( ) :
returns a list of the numbers in the range start…stop inclusive.

A boolean function prime_or_not( ) :
checks whether the input number is a prime number.

The function prime( ) :
print the result.

▷執行結果：

```
liumengyunde-MacBook-Pro:hw1 newmileou$ sbcl --script problem1_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

```
;;;file: problem1_2.lsp
;;;determine whether its contents form a palindrome

(defun palindrome_or_not(content)
    (equal content (reverse content)))

(defun palindrome(c)
    (if (palindrome_or_not c)
        (format t "~A form a palindrome.~%" c)
        (format t "~A do not form a palindrome.~%" c)
    )
)

(palindrome '(a b c ))
(palindrome '(m a d a m))
(palindrome '(cat dog))
(palindrome '())
(palindrome '(cat dog bird bird dog cat))
```

A boolean function palindrome_or_not( ) :
if parameter content form a palindrome(equal to its reverse content) then the function returns true.

The function palindrome( ) :
if palindrome_or_not( ) return true —> print content form a palindrome.
if palindrome_or_not( ) return false —> print content do not form a palindrome.

▷執行結果：

```
liumengyunde-MacBook-Pro:hw1 newmileou$ sbcl --script problem1_2.lsp
(A B C) do not form a palindrome.
(M A D A M) form a palindrome.
(CAT DOG) do not form a palindrome.
NIL form a palindrome.
(CAT DOG BIRD BIRD DOG CAT) form a palindrome.
```

◎Problem 1.3

```
;;;file: problem1_3.lsp
;;;Fibonacci function with original recursion and tail recursion


;;Original func. name : fib1
(princ "Original recursion:")
(format t "~%")
(defun fib1(n)
    "Compute the n'th Fibonacci number."
    (if (< n 2)
        n
        (+ (fib1 (- n 1)) (fib1 (- n 2)))))

(trace fib1)
(format t "~D~%~%" (fib1 3))

;;Tail func. name : fib2
(princ "Tail recursion:")
(format t "~%")
(defun fib_nto2(a b n)
     (if (= n 0)
          a
          (fib_nto2 b (+ a b) (- n 1))))
(defun fib2(n)
    (fib_nto2 0 1 n))

(trace fib2)
(format t "~D~%" (fib2 8))
```

The original Fibonacci function fib1( ):
if n<2 —> return n值
else —> return fib1(n-1)+fib1(n-2)

Use trace to show the execution details of (fib1 3) and use format to print its answer.

A tail-recursive function is one in which the recursive call occurs last .
Use fib2 to call the recursive-function fib_nto2.

Use trace to show the execution details of (fib2 8) and use format to print its answer.

▷執行結果：

```
liumengyunde-MacBook-Pro:hw1 newmileou$ sbcl --script problem1_3.lsp
Original recursion:
  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

Tail recursion:
  0: (FIB2 8)
  0: FIB2 returned 21
21
```

◎Problem 3

```lisp
;;;file: diff.lsp
;;;diff is an useful command on Unix-based system. You can use it to compare two files. It will show you the difference between them.

(defvar list1 nil)
(defvar list2 nil)

;; check if the element in list1 is equal to one element in list2
(defun same(element1)
    (dolist (element2 list2)
        (if (equal element1 element2)
            (return-from same T)))
    (return-from same nil))

;; read file1
(with-open-file
    (stream "file1.txt"
        :direction :input
        :if-does-not-exist nil)
    (loop for line1 = (read-line stream nil 'eof) until (eq line1 'eof)
        do (push line1 list1)))

;; read file2
(with-open-file
    (stream "file2.txt"
        :direction :input
        :if-does-not-exist nil)
    (loop for line2 = (read-line stream nil 'eof) until (eq line2 'eof)
        do (push line2 list2)))

;; compare two files
(setq list1 (reverse list1))
(setq list2 (reverse list2))
(dolist (element1 list1)
    (if (eq (same element1) T)
        (dolist (element2 list2)
            (if (equal element1 element2)
                ;; true:print same lines
                (progn
                    (format t " ~A~%" (pop list1))
                    (pop list2)
                    (return))
                ;; false:print c++ code
                (format t "~c[32m+~A~c[0m~%" #\ESC (pop list2) #\ESC)))  ;"~c[32mabc~c[0m~%" ---> green"abc"
        (format t "~c[31m-~A~c[0m~%" #\ESC (pop list1) #\ESC)))  ;red
```

A boolean function same( ) :
if element1 equal to element2 –> return true
else –> return false
之後用於比較file1.txt & file2.txt

Use with-open-file to read the files.
一行一行去讀取 之後再push進list中
將file存取成list的形式
之後還需reverse list 才是file的原貌

Compare two files part:

1. element1 represents file1
   element2 represents file2
2. Basic concept : use one of the line in file1 to compare with the whole file2.
3. if (equal element1 element2) return true :
   represent that file1 & file2 have the same line then we print it and pop the same line out of the list2.
   else :
   print the difference contents.
4. (format t "~c[32mtext~c[0m") will print the green text.
   (format t "~c[31mtext~c[0m") will print the red text.

▷執行結果：

```
liumengyunde-MacBook-Pro:hw1 newmileou$ sbcl --script diff.lsp
-#include <stdio.h>
+#include <iostream>
+using namespace std;
 int main() {
-  printf("Hello World");
+  cout << "Hello World" << end;
   return 0;
 }
```