題目 1-1：

執行環境：mac terminal (using vscode)

執行步驟：sbcl --script problem1-1.lsp

程式碼說明：

利用 for 迴圈從 i=2 到(i*i >= input_number)去取 input_number / i 的餘數，若餘數等於 0，則代表他不是質數而跳出。

```
(defun prime (input_number) ; 定義 input_number
    (let ((i 2))  ;令 i=2
        (loop
            (cond ((> (* i i) input_number) ; if (i*i >= input_number)
                (format t "True~%")
                (return)))
            (cond ((= (mod input_number i) 0) ; 取 input_number / i 的餘數, 若餘數等於 0
                (format t "False~%")
                (return)
                ))
            (setq i (+ i 1))
        )
    )
)
```

題目 1-2：

執行環境：mac terminal (using vscode)

執行步驟：sbcl --script problem1-2.lsp

程式碼說明：

去 reverse input_list，比較原 list 跟反過來的 list 是否相同即可。

```
(defun palindrome (input_list)
    (let ((tmpList (reverse input_list)))    ; reverse input_list 到 tmplist
        (if (equal tmpList input_list)  ; 如果相等就輸出 true，用 equal 去比較數值而已
            (format t "True~%")
            (format t "False~%")
        )
    )
)
```

題目 1-3：

執行環境：mac terminal (using vscode)

執行步驟：sbcl --script problem1-3.lsp

程式碼說明：

第一個是普通的迭代，第二個有回傳一個數值回去，我覺得有點像 while 迴圈去做 i-1 的條件而已，只是用迭代去實現。

```
(defun fib1 (n)
    (if (<= n 2)  ;如果 n<=2 回傳 1，否則回傳 fib1(n-1)+fib1(n-2)
        1
        (+ (fib1 (- n 1)) (fib1 (- n 2)))
    )
)

(defun fib2 (n)
    (defun tail-fib (n a b)     ; 因為要在一個 defun 內做 recursive，因此只能用 labels
            (if (= n 0)    ; n 是要找的第幾個數，a 是 a1，b 是 a2，最後的結果會存在 a 內
                a
                (tail-fib (- n 1) b (+ a b))
            )
    )
    (tail-fib n 0 1);第一次迭代
)
(trace fib1)
(trace fib2)
```

題目 2：

執行環境：mac terminal (using vscode)

執行步驟：sbcl --script mergesort.lsp

程式碼說明：

第一段就是單純的 mergesort，利用資結學過的 recursive 版的 mergesort，先寫出 rmergeSort，再寫出 listMerge 即可實現 mergesort。

第二段照著作業需求去應用 mergesort。

```
(defun merge_sort(lst)
    (defun merge_(f s)
        (cond
        ((= (list-length f) 0) s)
        ((= (list-length s) 0) f)
        ((< (car f) (car s)) (append (list (car f)) (merge_ (cdr f) s)))
        ((> (car f) (car s)) (append (list (car s)) (merge_ f (cdr s))))
        ((= (car f) (car s)) (append (list (car f) (car s)) (merge_ (cdr f) (cdr s))))
        )
    )
    (let ((len (list-length lst)))
        (cond
        ((= len 1) lst)
        (t
            (merge_ (merge_sort (subseq lst 0 (ceiling (/ len 2))))
                    (merge_sort (subseq lst (ceiling (/ len 2))))
            )
        )
        )
    )
)
```

```
(let ((x 0) (tmplist ()))
    (setq x (read))   ; 輸入要輸入幾個
    (dotimes (i x)    ; 要做幾次
        (let ((tmp 0)); 把輸入得值存在 list 內
            (setq tmp (read))
            (push tmp tmplist)
        )
    )
    (setq tmplist (reverse tmplist));因為用 push 所以要 reverse 回來
    (dolist (n (merge_sort tmplist));輸出 mergesort 跑完得 list 出來
        (format t "~d " n))
    (format t "~%")
)
```

題目 3：

執行環境：mac terminal (using vscode)

執行步驟：sbcl --script diff.lsp

程式碼說明：

程式碼主要分成三部分，第一部分用於取出 file 內容，並逐行以一維 list 形式存在 getIn1 以及 getIn2 中。第二部分是一個函數（get_list_item），第一個參數=1 指定要存取 getIn1，第二個參數是要取出指定 list 中的第幾個。第三部分是主程式的部分，假設 hello-world.c 是 file1.txt，hello-world.cpp 是 file2.txt，則中心思想是先用 file1 的第一行去比較 file2 的所有行，如果都沒有相同的，則輸出"- <此行內容>"，並取用 file1 下一行再去比，如果在 file2 第 5 行找到相同的行，則輸出 1~4 行的"+ <此行內容>"，而輸出第 5 行的"<此行內容>"，並下一次比較 file2 的行數更新從第 6 行開始，以此類推，則能搜尋完全部，但最後一行"}"要補上。

顏色輸出：(format t "~c[31m <想顯示的內容> [0m~%" #\ESC <變數>#\ESC)，而引號其中的 31 跟 0，分別代表顏色顯示紅色，背景不變，若要顯示綠色則將 31 改成 32。

```
;;;;;;;; Get All Needed Variable ;;;;;;;;
(defvar in1 (open "./res/file1.txt" :if-does-not-exist nil)) ;將 file1 內容存進去 in1
(defvar countIn1 0) ;用於計算 file1 行數
(defvar getIn1 ())  ;用於將內容以一維 list 形式存在 getIn1

(loop for line = (read-line in1 nil)  ;迴圈逐行取出 in1 並暫存在 line 中
   while line do (setq getIn1 (push line getIn1)) ;將 line 存到 getIn1 中
                 (setq countIn1 (+ countIn1 1)) ;順便使 countIn1++以計算行數
)
(setq getIn1 (reverse getIn1)) ;由於是用 push，因此結果要用 reverse 轉回來

;以下都跟 file1 相同
(defvar in2 (open "./res/file2.txt" :if-does-not-exist nil))
(defvar countIn2 0)
(defvar getIn2 ())
(loop for line = (read-line in2 nil)
   while line do (setq getIn2 (push line getIn2))
                 (setq countIn2 (+ countIn2 1))
)
(setq getIn2 (reverse getIn2))
(close in1) ;關閉 open 指標
```

```
(close in2) ;關閉 open 指標
;;;;;;;; Utils ;;;;;;;;
(defvar return_get_list_item "") ;由於不知道怎麼用 return，因此設一個變數當作是 return value
(defun get_list_item(list_number number) ;number 代表要取出 list 中的第幾個 element
    (let ((tmplist1 ())) ;宣告一個 local 變數來做操作
        (if (equal 1 list_number) ; list_number=1 代表要取用 getIn1
            (setq tmplist1 getIn1)
            (setq tmplist1 getIn2)
        )
        (setq return_get_list_item (car tmplist1)) ;取出第一個值並存在 return value 中
        (setq number (- number 1)) ;number 數量-1
        (if (> number 0) ;如果 number 一開始>1，就會往下做
            (loop
                (setq tmplist1 (cdr tmplist1)) ;取出剩下的放進 tmplist1
                (setq return_get_list_item (car tmplist1));取出第一個值放 return value 中
                (setq number (- number 1)) ;number 數量-1
                (when (equal number 0) (return)) ;一直重複直到 number==0，就可取到要求位置的值
            )
        )
    )
)

;;;;;;;; Main Function ;;;;;;;;
(defvar file2Line 1) ;紀錄 file2 要比較的位置的頭

(let ((i 1) (j 1)) ;宣告一個 i=1, j=1
    (loop
        (get_list_item 1 i) ;取出第一個 list 的第 i 個行數內容
        (let ((return_list1 return_get_list_item)) ;將取出值存在 return_list1
            (block inner_loop
                (loop
                    (get_list_item 2 j) ;取出第二個 list 的第 j 個行數內容
                    (cond ((equal return_list1 return_get_list_item) ;比較左右行數內容
                           (loop  ;如果相同
                               (get_list_item 2 file2Line) ;取出上一個比較的位置內容
```

```
                                (format t "~c[32m+ ~a~c[0m~%" #\ESC
return_get_list_item #\ESC) ;輸出綠色成”+ <想顯示的內容>”
                                (setq file2Line (+ 1 file2Line)) ;一直輸出直到上一個比較的位
置等於目前相等的位置
                                (cond ((equal file2Line j) ;一但相等
                                    (get_list_item 2 file2Line) ;取出相等位置內容
                                    (write-line return_get_list_item) ;正常輸出
                                    (setq file2Line (+ 1 file2Line)) ;使上一個比較的位置更新
到下一個位置
                                    (return-from inner_loop 0)) ;從最大的 loop 離開
                                )
                            )
                        )
                    )
                    (setq j (+ 1 j))  ;若沒有相同就會跳進來這裡繼續比較 file2 下一行
                    (when (equal j countIn2) (return)) ;若比較到文末沒有相等一樣跳出迴圈
                )
            )
            (cond ((equal j countIn2) ;若比較到文末沒有相等
                (format t "~c[31m- ~a~c[0m~%" #\ESC return_list1 #\ESC) ;輸出紅色成”-
<想顯示的內容>”
                )
            )
            (setq j file2Line) ;使 j 等於上一個比較的位置
        )
        (setq i (+ 1 i)) ;選擇 file1 下一行繼續比較
        (when (equal i countIn1) (return)) ;比較到 file1 文末則比較結束
    )
)
(format t "}~%") ;補上最後一個括號
```