Take a look at this implementation provided by Tony Ha.

Everything should be clear except what template he’s used.

What is the purpose of the template?

Why do we need one?

And what template can we rely on below:

; sort1 : ListOfNumber -> ListOfNumber
; implements one pass through the list
(define (sort1 lon)
  (cond ((empty? lon) lon)
        ((empty? (rest lon)) lon) ; (cons (first lon) empty))
        ((<= (first lon) (second lon)) (cons (first lon) (sort1 (rest lon))))
        ((< (second lon) (first lon)) (cons (second lon) (sort1 (remove (second lon) lon))))))

(check-expect (sort1 (list 4 3 5 2 6 1)) (list 3 4 2 5 1 6))

; sort2 : ListOfNumber -> ListOfNumber
; runs passes on the input until sorted
(define (sort2 lon)
  (cond ((empty? lon) empty)
        ((equal? lon (sort1 lon)) lon)
        (else (sort2 (sort1 lon))))))