; ● CSC104 Winter 2020 — Exercise #5 — Print out and fill in by hand, then hand in to the TA at the start of your quiz. ●

; UTorID (login ID) :

; Surname :

; Given Name :

; Assume the following three definitions ...

```
(define list.A (list 1 (list (list 2 3) 4 5) (list) (list 6 (list 7 8))))
(define list.B (list (list 1 (list) 2 (list 3)) 4 (list (list 5 6 7) 8)))
(define list.C (list (list (list 1) (list 2 3) (list)) (list 4 5) (list (list 6 7)) 8))
```

; Here's the literal for list.A again, but with a box drawn around each element : (list [1] [(list (list 2 3) 4 5)] [(list)] [(list 6 (list 7 8))]) .

; Write out the literals for list.B and list.C again, with a box drawn around each element ...

; Here's a simple assertion, using the literal expression for the value of list.A again, but reformatted so that each element of each list is on its own line ...

```
(same! list.A (list 1
                    (list (list 2
                                3)
                          4
                          5)
                    (list)
                    (list 6
                          (list 7
                                8))))
```

; ... which is what the "Stack  " button does in DrRacket, if you click at the beginning of the list literal and then click that button in the toolbar.

; Do that for the literal expressions for list.B and list.C ...

```
(same! list.B
```

```
(same! list.C
```

; Beside each of the following expressions show its value ...

```
(length list.A)

(length list.B)

(length list.C)

(length (list list.A list.B list.C))

(map length (list list.A list.B list.C))
```

; Beside each of the following expressions show its value ...

```
(reverse list.A)
```

```
(reverse list.B)
```

```
(reverse list.C)
```

; Show, with standard underlining, the steps for : `(step (map reverse (rest (list 1 (list (list 2 3) 4 5) (list) (list 6 (list 7 8))))))`

; Assume the following definition has been entered/run ...

```
(define (maybe-length v)
  (if (list? v) (length v)
      else __)) ; that "__" is an image, from (solid-rectangle 8 1)
```

; ... then show the steps, with standard underlining, for ...

```
(step (maybe-length "coffee"))
```

```
(step (maybe-length (list "coffee" "tea" "water")))
```

```
(step (hide maybe-length) (map maybe-length (list "coffee" "tea" "water")))
```

```
(step (hide maybe-length) (map maybe-length (list 1 (list (list 2 3) 4 5) (list) (list 6 (list 7 8)))))
```

```
(step (hide maybe-length) (map maybe-length (list (list 1 (list) 2 (list 3)) 4 (list (list 5 6 7) 8))))
```

```
(step (hide maybe-length) (map maybe-length (list (list (list 1) (list 2 3) (list)) (list 4 5) (list (list 6 7)) 8)))
```