

Homework 02

● Graded

Student

Giancarlos Marte

Total Points

94 / 94 pts

Autograder Score

89.0 / 89.0

Passed Tests

Exercise 1. pair (0/0)

Exercise 1. pair-left (1/1)

Exercise 1. pair-right (1/1)

Exercise 1.a. pair-set-right (3/3)

Exercise 1.b. pair-set-left (3/3)

Exercise 1.c. pair-swap (3/3)

Exercise 1.d. pair-add (4/4)

Exercise 2.b. first-name (4/4)

Exercise 2.c. last-name (4/4)

Exercise 2.d. full name (4/4)

Exercise 2.e. initials (5/5)

Exercise 3. max-from (5/5)

Exercise 4. min-from (5/5)

Exercise 6. count (11/11)

Exercise 7. sum (12/12)

Exercise 8. occurrences (12/12)

Exercise 9. norm (12/12)

Question 2

Adjustments

0 / 0 pts

✓ - 0 pts No adjustments

- 5 pts (2) Representation is not a function (`(lambda)`)

- 0 pts (2) Only `first-name` and `last-name` need to be "methods"

- 1 pt (9) should use `(map sqr _)` and `sum` (from exercise 7)

- 5 pts Late penalty

Question 3

Problem 5

5 / 5 pts

✓ - 0 pts Correct

- 5 pts Missing

- 4 pts Incomplete or doesn't work

- 2 pts Did not redefine min-from, max-from

Autograder Results

Exercise 1. pair (0/0)

Exercise 1. pair-left (1/1)

Exercise 1. pair-right (1/1)

Exercise 1.a. pair-set-right (3/3)

Exercise 1.b. pair-set-left (3/3)

Exercise 1.c. pair-swap (3/3)

Exercise 1.d. pair-add (4/4)

Exercise 2.b. first-name (4/4)

Exercise 2.c. last-name (4/4)

Exercise 2.d. full name (4/4)

Exercise 2.e. initials (5/5)

Exercise 3. max-from (5/5)

Exercise 4. min-from (5/5)

Exercise 6. count (11/11)

Exercise 7. sum (12/12)

Exercise 8. occurrences (12/12)

Exercise 9. norm (12/12)

Submitted Files

```
1  #lang racket
2  (require rackunit)
3  #|
4      #####
5      ### PLEASE DO NOT DISTRIBUTE SOLUTIONS PUBLICLY ###
6      #####
7  |#
8  (provide (all-defined-out))
9  ;; ^^^^^ DO NOT CHANGE ANY CODE ABOVE THIS LINE ^^^^^
10
11  ::::::::::::::::::::::::::::::::::::::::::::
12  ;; Exercise 1
13
14  ;; point, point?, point-left, point-right
15  ;; pair: Symbol Symbol
16  (struct pair (left right) #:transparent)
17
18  #|(define pair 'delete-me-after-defining-your-struct)
19  (define pair-left 'delete-me-after-defining-your-struct)
20  (define pair-right 'delete-me-after-defining-your-struct)
21  |#
22
23  ;; Exercise 1.a
24  ;; pair-set-left: Pair Symbol -> Pair
25  (define (pair-set-left p l)
26    (match p
27      [(pair lh rh)
28       (pair l rh)]))
29
30  ;; Exercise 1.b
31  ;; pair-set-right: Pair Symbol -> Pair
32  (define (pair-set-right p r)
33    (match p
34      [(pair lh rh)
35       (pair lh r)]))
36
37  ;; Exercise 1.c
38  ;; pair-swap: Pair -> Pair
39  (define (pair-swap p)
40    (match p
41      [(pair lh rh)
42       (pair rh lh)]))
43
44  ;; Exercise 1.d
45  ;; You can only use match* one time. You cannot use match.
46  ;; pair-add: Pair Pair -> Pair
```

```

47 (define (pair-add p1 p2)
48   (match* (p1 p2)
49     [((pair lh rh) (pair lh2 rh2)) (pair (+ lh lh2) (+ rh rh2))]))
50
51 .....
52 ;; Exercise 2.a
53 ;; name: String String -> (X -> String)
54 (define (name first last)
55   (lambda (m) (m first last)))
56
57 ;; Exercise 2.b
58 ;; first-name: Name -> String
59 (define (first-name p)
60   (p (lambda (f l) f)))
61
62 ;; Exercise 2.c
63 ;; last-name: Name -> String
64 (define (last-name p)
65   (p (lambda (f l) l)))
66
67 ;; Exercise 2.d
68 ;; full-name: Name -> String
69 (define (full-name p)
70   (p (lambda (f l)
71       (string-append (string-append f " ") l))))
72
73 ;; Exercise 2.e
74 ;; initials: Name -> String
75 (define (initials p)
76   (p (lambda (f l)
77       (string-append
78         (substring f 0 1) (substring l 0 1)))))
79
80 .....
81 #|
82 ;; Exercise 3
83 ;; max-from: Real (Listof Real) -> Real
84 (define (max-from n l)
85   (match l
86     ['() n]
87     [(cons f r) (max-from (max n f) r))])
88
89 .....
90 ;; Exercise 4
91 ;; min-from Real (Listof Real) -> Real
92 (define (min-from n l)
93   (match l
94     ['() n]
95     [(cons f r) (min-from (min n f) r))])

```

```

96 |#
97
98 ::::::::::::::::::::::::::::::::::::
99 ;; Exercise 5: revisit Exercise 3 and Exercise 4
100
101 ;; Exercise 3
102 ;; max-from: Real (Listof Real) -> Real
103 (define (max-from n l)
104   (from n l max-from max))
105
106 ;; Exercise 4
107 ;; min-from Real (Listof Real) -> Real
108 (define (min-from n l)
109   (from n l min-from min))
110
111 ;; auxiliary
112 ;; from: Real (Listof Real) (Real (Listof Real) -> Real) (Real Real -> Real) -> Real
113 (define (from n l mainf innerf)
114   (match l
115     ['() n]
116     [(cons f r) (mainf (innerf n f) r))]))
117
118 ::::::::::::::::::::::::::::::::::::
119 ;; Exercise 6
120 ;; count: (Listof Real) -> Real
121 (define (count l)
122   (match l
123     ['() 0]
124     [(cons f r) (+ 1 (count r))]))
125
126 ::::::::::::::::::::::::::::::::::::
127 ;; Exercise 7
128 ;; sum: (Listof Real) -> Real
129 (define (sum l)
130   (match l
131     ['() 0]
132     [(cons f r) (+ f (sum r))]))
133
134 ::::::::::::::::::::::::::::::::::::
135 ;; Exercise 8
136 ;; occurrences: Symbol (Listof symbol) -> Real
137 (define (occurrences x l)
138   (match l
139     ['() 0]
140     [(cons f r)
141      (cond
142        [(equal? f x) (+ 1 (occurrences x r))]
143        [else (+ 0 (occurrences x r))])]))
144

```

```
145 .....  
146 ;; Exercise 9  
147 ;; norm: (Listof Real) -> Real  
148 (define (norm l)  
149   (define addSqr (sum(map sqr l)))  
150   (sqrt addSqr))  
151  
152  
153
```