```
1: $Id: solution-2017q4-midterm.txt,v 1.5 2017-11-08 13:01:48-08 - - $
 2: Solution to cmps112-2017q4-midterm, page 1
 3:
 4:
 5: Question 1(a). [2]
 6:
7: let sum list =
        let rec sum' list acc = match list with
8:
9:
            | [] -> acc
10:
            | x::xs \rightarrow sum' xs (x + acc)
11:
        in sum' list 0
12: ... deduct 1 point if correct, but not tail recursive
13:
14:
15: Question 1(b). [2]
17: let rec fold_left fn unit list = match list with
18:
        | [] -> unit
19:
        | x::xs -> fold_left fn (fn unit x) xs
20: ... deduct 1 point if correct, but not tail recursive
21:
22:
23: Question 1(c). [2]
24:
25: let sumf = fold_left (+) 0
26:
27:
28: Question 2. [2]
29:
30: (define (reverse list)
            (define (rev in out)
32:
                     (if (null? in) out
33:
                         (rev (cdr in) (cons (car in) out))))
            (rev list '()))
35: ... deduct 1 point if correct, but not tail recursive
36: ALTERNATE:
37: (define (reverse list)
            (foldl (lambda (a d) (cons a d)) '() list))
39: ... add 1 bonus point if uses foldl, and if used CORRECTLY.
40:
41:
42: Question 3. [2]
43:
44: (define (map f list)
            (if (null? list) '()
45:
46:
                (cons (f (car list)) (map f (cdr list)))))
47:
```

```
48:
49: Solution to cmps112-2017q4-midterm, page 2
51:
52: Question 4. [2]
53:
54: fac
                                       Grading:
           int -> int
55: n
                                       9 correct -> 2 points
           int
56: fac'
           int -> int -> int
                                      8 or 7 correct -> 1.5 points
                                      6 or 5 correct -> 1 point
57: n'
           int
58: a'
           int
                                      4 or 3 correct -> 0.5 points
59: <=
          'a -> 'a -> bool
                                      2 or fewer correct -> 0 points
60: 1
           int
 61: -
           int -> int -> int
62: *
           int -> int -> int
63:
64:
65: Question 5(a). [2]
67: (define (sum list)
             (define (summ list acc)
68:
69:
                     (if (null? list) acc
70:
                          (summ (cdr list) (+ (car list) acc))))
71:
             (summ list 0))
72: ... deduct 1 point if correct, but not tail recursive
73:
74:
75: Question 5(b). [2]
77: (define (fold_left fn unit list)
             (if (null? list) unit
79:
                 (fold_left fn (fn unit (car list)) (cdr list))))
80: ... deduct 1 point if correct, but not tail recursive
81:
82:
83: Question 5(c). [2]
84:
85: (define (sumf list)
86:
             (fold_left + 0 list))
87:
88:
89: Question 6(a). [2]
90:
91: let rec evenlen list = match list with
       | [] -> true
92:
93:
         | [_] -> false
94:
         | car::cadr::cddr -> evenlen cddr
95:
97: Question 6(b). [2]
99: let evenlen = List.fold_left (fun t _ -> not t) true
100:
```

```
101:
102: Solution to cmps112-2017q4-midterm, page 3
104:
105: Question 7. [2]
106:
107: universal - parametric (or template or generic)
108: universal - inclusion (or oop)
109: ad hoc - conversion
110: ad hoc - overloading
111: ... assign 1/2 point for each pair (left and right column)
112: ... that are correct
113:
114:
115: Question 8. [2]
117: let reverse list =
118:
         let rec rev inl outl = match inl with
119:
             | [] -> outl
120:
             | x::xs -> rev xs (x::outl)
121:
         in rev list []
122: ... deduct 1 point if correct, but not tail recursive
123: ALTERNATE:
124: let reverse = List.fold_left (fun t h -> h::t) [];;
125: ... add 1 bonus point if uses foldl, and if used CORRECTLY.
126:
127:
128: Question 9. [2]
129:
130: node reverse (node head) {
131: node out = null;
       while (head != null) {
132:
         node t = head;
133:
134:
          head = head.link;
135:
          t.link = out;
136:
           out = t;
137:
        }
138: }
139:
140:
141: Question 10. [4]
142:
143: let collatz n =
         let rec collatz' n rest =
144:
            if n <= 1
145:
146:
                then 1::rest
                else if n \mod 2 = 0
147:
148:
                        then collatz' (n / 2) (n::rest)
149:
                        else collatz' (n * 3 + 1) (n::rest)
        in List.rev (collatz' n [])
151: ... deduct 1 point if correct, but collatz fn not tail recursive
152:
```

11/08/17 13:01:48

## \$cmps112-wm/TA/.admin/midterm/solution-2017q4-midterm.txt

4/4

```
153:
154: Solution to cmps112-2017q4-midterm, page 4
155:
            (C) \lambda-calculus
156:
      1.
157:
           (A) strong and dynamic
158:
      2.
159:
      3.
            (B) strong and static
160:
161:
            (A) ALGOL 60
162:
      4.
163:
164:
      5.
            (D) $ 0 ( 2 sup n ) $
165:
            (A) $ O (n) $
166:
167:
            (A) (apply + '(1 2 3 4))
168:
      7.
169:
     8.
            (B) Edsger Dijkstra
170:
171:
            (D) only M, but neither D nor U.
172:
     9.
173:
174: 10.
            (D) int -> int -> int
175:
            (D) (cddr '(1 2 3 4))
176: 11.
177:
            (B) $x$ is bound and $y$ is free.
178: 12.
179:
```