```
$Id: 2017q2-final.txt,v 1.2 2017-06-14 17:37:40-07 - - $
Answers to cmps112-2017q2-final, page 1

_____________________________________________________________________________
Question 1. [1][1]
Any order is acceptable:
edge(1,2).
edge(1,3).
edge(2,3).
edge(2,5).
edge(3,4).
adjacent(X,Y) :- edge(X,Y).
adjacent(X,Y) :- edge(Y,X).

_____________________________________________________________________________
Question 2. [2]
fib      int -> int                    (1/2 pt for fib, fib')
fib'     int -> int -> int -> int
n        int                           (1/2 pt for each int)
m        int
a        int
b        int
-        int -> int -> int             (1/2 pt for - and +)
+        int -> int -> int
=        'a -> 'a -> bool              (1/2 pt for = and <)
<        'a -> 'a -> bool
failwith string -> 'a

_____________________________________________________________________________
Question 3. [2][1]
let car list = match list with        (1 pt for car)
  | [] -> failwith "car []"
  | x::_ -> x;;
let cdr list = match list with        (1 pt for cdr)
  | [] -> failwith "cdr []"
  | _::xs -> xs;;
car : 'a list -> 'a = <fun>           (1/2 pt for each type spec)
cdr : 'a list -> 'a list = <fun>

_____________________________________________________________________________
Question 4. [1][1]
sum([],0).
sum([H|T],S) :- sum(T,U), S is U + H.
lengthh([],0).
lengthh([H|T],L) :- lengthh(T,U), L is U + 1.

_____________________________________________________________________________
Question 5. [1]
[1:-]--->[2:-]--->[|:-]-------------->[|:-]--->[7:o]
                   V                   V
                  [3:-]--->[4:o]      [|:-]--->[6:o]
                                       V
                                      [5:o]
```

```
54:
55: Answers to cmps112-2017q2-final, page 2
56:
57: ____________________________________________________________________________
58: Question 6. [2]
59:
60: C++       |Bjarne Stroustrup's most noted contribution.
61: COBOL     |Business data processing language, Grace Hopper.
62: LISP      |List processing with Lots of Idiotic Silly Parentheses.
63: FORTRAN   |Numeric and scientific computation language developed at IBM.
64: Simula 67|Simulation language that influenced the design of C++.
65: Pascal    |Small language for structured programming designed by Niklaus Wirth.
66:
67: ____________________________________________________________________________
68: Question 7. [4]
69:
70: Array extend [
71:    max [
72:       (self size = 0)
73:       ifTrue: [ ^ nil ]
74:       ifFalse: [ |mx|
75:          mx := self at: 1.
76:          2 to: self size do: [ :i |
77:             ((self at: i) > mx) ifTrue: [mx := self at: i].
78:          ].
79:          ^ mx.
80:       ]
81:    ]
82: ].
83:
84: ____________________________________________________________________________
85: Question 8. [2]
86:
87: (define (reverse list)
88:         (define (rev in out)
89:                 (if (null? in) out
90:                     (rev (cdr in) (cons (car in) out))))
91:         (rev list '()))
92:
93: ____________________________________________________________________________
94: Question 9. [2]
95:
96: let sum = List.fold_left (+) 0;;
97: let length = List.fold_left (fun n _ -> n + 1) 0;;
98:
```

```
 99:
100: Answers to cmps112-2017q2-final, page 3
101:
102: ____________________________________________________________________________
103: Question 10. [1]
104:
105: let twice f x = f (f x);;
106:
107: ____________________________________________________________________________
108: Question 11. [2]
109:
110: father( F, C) :- parents( F, _, C).
111: mother( M, C) :- parents( _, M, C).
112:
113: ____________________________________________________________________________
114: Question 12. [1][1][1]
115:
116: (define (oddlen list)
117:         (cond ((null? list) #f)
118:               ((null? (cdr list)) #t)
119:               (else (oddlen (cddr list)))))
120:
121: let rec oddlen list = match list with
122:     | [] -> false
123:     | [_] -> true
124:     | x::y::tail -> oddlen tail
125:
126: oddlen([_]).
127: oddlen([H1,H2|T]) :- oddlen(T).
128:
129: ____________________________________________________________________________
130: Question 13. [2][2]
131:
132: let rec map f lis = match lis with
133:     | [] -> []
134:     | x::xs -> f x :: map f xs;;
135:
136: (define (map f lis)
137:         (if (null? lis) '()
138:             (cons (f (car lis)) (map f (cdr lis)))))
139:
```

```
140:
141: Answers to cmps112-2017q2-final, page 4
142:
143:  1.    (C) $ O ( n ) $
144:
145:  2.    (A) $ O ( 1 ) $
146:
147:  3.    (D) - : int -> int -> int = <fun>
148:
149:  4.    (D) vector
150:
151:  5.    (C) A structure on the heap, used to hold variables of an outer
152:             function when referenced by an inner function.
153:
154:  6.    (C) $ O ( n ) $
155:
156:  7.    (D) (cddr '(1 2 3 4))
157:
158:  8.    (C) [1 + 4] value.
159:
160:  9.    (A) ((2*3)+4)*5
161:
162: 10.    (A) strong and dynamic
163:
164: 11.    (A) (apply * '(1 2 3 4))
165:
166: 12.    (C) Edsger W. Dijkstra
167:
```

```
168:
169: Answers to cmps112-2017q2-final, page 4
170:
171:  1.    (C) monad
172:
173:  2.    (D) Smalltalk
174:
175:  3.    (C) a+
176:
177:  4.    (C) $x = "$foo\n";
178:
179:  5.    (B) c()
180:
181:  6.    (C) 'a list -> 'a
182:
183:  7.    (C) universal inclusion
184:
185:  8.    (B) ad hoc overloading
186:
187:  9.    (D) universal parametric
188:
189: 10.    (A) ad hoc conversion
190:
191: 11.    (D) throw
192:
193: 12.    (A) Algol 60
194:
195:
```