

```
1: $Id: 2016q4-final,v 1.2 2016-12-06 12:03:00-08 - - $
2: Answers to cmps112-2016q4-final, page 1
3:
4:
5: Question 1. [3]
6:
7: C++          |Bjarne Stroustrup's noted contribution to language design.
8: COBOL        |Business data processing language, designers Grace Hopper.
9: Algol 60     |Designed in Europe to express algorithms in a structured way.
10: C            |First version of Unix was 9000 lines of this language.
11: Lisp         |List processing language used in artificial intelligence.
12: FORTRAN      |Numeric and scientific computation language developed at IBM.
13: Simula 67    |Simulation language that influenced the design of C++.
14: Pascal       |Small language for structured programming by Niklaus Wirth.
15: Java        |Sun Micro claimed this language write once, run anywhere.
16:
17:
18: Question 2. [2]
19:
20: arrow(a,b) .
21: arrow(a,c) .
22: arrow(b,c) .
23: arrow(b,e) .
24: arrow(c,d) .
25: arrow(d,e) .
26:
27:
28: Question 3. [2]
29:
30: ispath(X,Y) :- arrow(X,Y) .
31: ispath(X,Y) :- arrow(X,Z) , ispath(Z,Y) .
32:
33:
34: Question 4. [3]
35:
36: findpath(X,Y,P) :- arrow(X,Y) , P=[X,Y] .
37: findpath(X,Y,P) :- arrow(X,Z) , findpath(Z,Y,Q) , P=[X|Q] .
38:
39: alternate:
40:
41: findpath(X,Y,[X,Y]) :- arrow(X,Y) .
42: findpath(X,Y,[X|Q]) :- arrow(X,Z) , findpath(Z,Y,Q) .
43:
```

```
44:
45: Answers to cmps112-2016q4-final, page 2
46:
47:
48: Question 5. [4]
49:
50: let max gt list = match list with
51:   | [] -> failwith "max"
52:   | x::xs -> let rec max' x xs = match xs with
53:               | [] -> x
54:               | y::ys -> if gt x y then max' x ys
55:                           else max' y ys
56:               in max' x xs
57: ;;
58:
59:
60: Question 6. [2]
61:
62: let rec zip x y = match x, y with
63:   | [], _ -> []
64:   | _, [] -> []
65:   | x::xs, y::ys -> (x,y)::zip xs ys
66:
67:
68: Question 7. [2]
69:
70: let rec unzip list = match list with
71:   | [] -> ([],[])
72:   | (a,b)::rest -> let (a1,b1) = unzip rest
73:                     in (a::a1, b::b1);;
74:
75:
76: Question 8. [2]
77:
78: gcd( X, Y, Z ) :- X > Y, T is X - Y, gcd( T, Y, Z ).
79: gcd( X, Y, Z ) :- X < Y, T is Y - X, gcd( X, T, Z ).
80: gcd( X, X, X ).
81:
```

```
82:
83: Answers to cmps112-2016q4-final, page 3
84:
85:
86: Question 9. [2]
87:
88: universal      parametric (template, generic)
89:                inclusion (inheritance, OO)
90:
91: ad hoc         conversion (coercion)
92:                overloading
93:
94:
95: Question 10. [2]
96:
97: (define (pairthem l1 l2)
98:   (if (or (null? l1) (null? l2)) '()
99:       (cons (list (car l1) (car l2))
100:             (pairthem (cdr l1) (cdr l2)))))
101:
102:
103: Question 11. [6]
104:
105: Object subclass: Stack [
106:   |array top|
107:   Stack class >> new [
108:     ^ Stack new: 10
109:   ]
110:   Stack class >> new: size [
111:     ^ super new init: size
112:   ]
113:   init: size [
114:     top := 0.
115:     array := Array new: size.
116:   ]
117:   pop [
118:     |result|
119:     result := array at: top.
120:     top := top - 1.
121:     ^ result.
122:   ]
123:   push: item [
124:     top := top + 1.
125:     array at: top put: item
126:   ]
127:   empty [
128:     ^ top = 0.
129:   ]
130: ]
131:
```

132:
133: Answers to cmpls112-2016q4-final, page 4
134:
135: 1. (A) APL
136:
137: 2. (B) Perl
138:
139: 3. (A) #!
140:
141: 4. (D) Lisp
142:
143: 5. (D) Simula 67
144:
145: 6. (B) 2.0 sqrt
146:
147: 7. (C) thunk
148:
149: 8. (B) 1958, John McCarthy.
150:
151: 9. (B) Ocaml
152:
153: 10. (B) $X = 1.2246467991473532e-16$
154:
155: 11. (B) duck-typing
156:
157: 12. (D) virtual function table
158:

```
159:
160: Answers to cmpls112-2016q4-final, page 4
161:
162: 1.      (C) A structure on the heap, used to hold variables of an outer
163:         function when referenced by an inner function.
164:
165: 2.      (B) Edsger Dijkstra
166:
167: 3.      (D) \w+
168:
169: 4.      (D) ? :
170:
171: 5.      (A) (apply + ' (1 2 3))
172:
173: 6.      (A) function call stack
174:
175: 7.      (D) throw
176:
177: 8.      (D) reference counting
178:
179: 9.      (B) race condition
180:
181: 10.     (D) val f : int -> int -> int -> int
182:
183: 11.     (A) (f ' ())
184:
185: 12.     (A) ('a -> 'b) -> 'a list -> 'b list
186:
```