```
1: $Id: 2012q1-soln2,v 1.1 2012-03-19 15:06:42-07 - - $
 2: Answers to 2012a1-test1, page 1
 3:
 4: Note: answers which are correct, but different from the key,
 5: still get full points.
 6:
7:
8: Question 1. [1]
9:
10: map f list = [f x | x < - list]
11:
12:
13: Question 2. [2]
15: let mapf fn list = fold_right (fun h t -> fn h :: t) list []
17:
18: Question 3. [2]
20: let rec mapr fn list = match list with
21:
        | [] -> []
        | h::t -> fn h :: mapr fn t
22:
23:
24:
25: Question 4. [2]
26:
27: let car list = match list with
        | [] -> failwith "car []"
28:
        | h::_ -> h
29:
30: let cdr list = match list with
       | [] -> failwith "cdr []"
        | _::t -> t
32:
33:
34:
35: Question 5. [3]
36:
37: \$0 = "s|.*/||;
38: my $status = 0;
39: my %hash;
40: for my $fname (@ARGV ? @ARGV : "-") {
41:
       open my $file, "<$fname"
42:
            or print STDERR "$0: $fname: $!\n" and $status = 1 and next;
       while (defined (my $line = <$file>)) {
43:
44:
          map {++$hash{$}_{}} split m/W+/, $line;
45:
          map \{++\$hash\{\$_{-}\}\}\ $line = m/(w+)/g;
46:
       }
47: }
48: map {print "$_ $hash{$_}\n"} sort keys %hash;
49: exit $status;
50:
```

```
51:
52: Answers to 2012a1-test1, page 2
54:
55: Question 6. [3]
56:
57: let zipwith f x 11 12 =
        let rec zipwith' 11 12 = match 11, 12 with
58:
59:
            | [], [] -> []
60:
            | [], h2::t2 -> f x h2 :: zipwith' [] t2
            | h1::t1, [] -> f h1 x :: zipwith' t1 []
61:
62:
            | h1::t1, h2::t2 -> f h1 h2 :: zipwith' t1 t2
63:
        in zipwith' 11 12
64:
65:
66: Question 7. [3]
67:
68: let max gt list = match list with
69:
        | [] -> None
70:
        | mx::t ->
71:
            let rec max' mx u = match u with
72:
                 | [] -> Some mx
73:
                 | h::t -> max' (if gt mx h then mx else h) t
74:
            in max' mx t
75:
76:
77: Question 8. [4]
78:
79: (define (zipwith f x 11 12)
80:
            (define (zip 11 12)
81:
                     (if (null? 11)
82:
                         (if (null? 12)
83:
                             ′()
84:
                              (cons (f x (car 12))
85:
                                    (zip '() (cdr 12))))
86:
                         (if (null? 12)
87:
                              (cons (f (car 11) x)
88:
                                    (zip (cdr l1) '()))
89:
                              (cons (f (car 11) (car 12))
90:
                                    (zip (cdr 11) (cdr 12))))))
91:
            (zip 11 12))
92:
```

03/19/12 15:06:42

## \$cmps112-wm/Old-Exams/.solutions/ 2012q1-test2.txt

3/3

```
93:
 94: Answers to 2012a1-test1, page 3
            (D) float -> float -> float
 96:
     1.
 97:
98:
     2.
            (C) List.fold_right
99:
            (A) ((3-4)/5)-6
100:
     3.
101:
102:
     4.
            (A) compose
103:
104:
     5.
            (C) A structure on the heap, used to hold variables of an outer
105:
                function when referenced by an inner function.
106:
            (B) $line = <$file>;
107:
     6.
108:
109:
     7.
           (C) thunk
110:
111:
            (B) int list
     8.
112:
113: 9.
            (B) access (static) link
114:
115: 10.
            (D) Smalltalk
116:
117: 11.
           (B) Edsger Dijkstra
118:
```