```
$Id: 2012q1-soln2,v 1.1 2012-03-19 15:06:42-07 - - $
Answers to 2012a1-test1, page 1

Note: answers which are correct, but different from the key,
still get full points.

_________________________________________________________________
Question 1. [1]

map f list = [f x | x <- list]

_________________________________________________________________
Question 2. [2]

let mapf fn list = fold_right (fun h t -> fn h :: t) list []

_________________________________________________________________
Question 3. [2]

let rec mapr fn list = match list with
    | [] -> []
    | h::t -> fn h :: mapr fn t

_________________________________________________________________
Question 4. [2]

let car list = match list with
    | [] -> failwith "car []"
    | h::_ -> h
let cdr list = match list with
    | [] -> failwith "cdr []"
    | _::t -> t

_________________________________________________________________
Question 5. [3]

$0 =~ s|.*/||;
my $status = 0;
my %hash;
for my $fname (@ARGV ? @ARGV : "-") {
   open my $file, "<$fname"
        or print STDERR "$0: $fname: $!\n" and $status = 1 and next;
   while (defined (my $line = <$file>)) {
      #map {++$hash{$_}} split m/\W+/, $line;
      map {++$hash{$_}} $line =~ m/(\w+)/g;
   }
}
map {print "$_ $hash{$_}\n"} sort keys %hash;
exit $status;
```

Answers to 2012a1-test1, page 2

Question 6. [3]

```
let zipwith f x l1 l2 =
    let rec zipwith’ l1 l2 = match l1, l2 with
        | [], [] -> []
        | [], h2::t2 -> f x h2 :: zipwith’ [] t2
        | h1::t1, [] -> f h1 x :: zipwith’ t1 []
        | h1::t1, h2::t2 -> f h1 h2 :: zipwith’ t1 t2
    in zipwith’ l1 l2
```

Question 7. [3]

```
let max gt list = match list with
    | [] -> None
    | mx::t ->
        let rec max’ mx u = match u with
            | [] -> Some mx
            | h::t -> max’ (if gt mx h then mx else h) t
        in max’ mx t
```

Question 8. [4]

```
(define (zipwith f x l1 l2)
        (define (zip l1 l2)
                (if (null? l1)
                    (if (null? l2)
                        ’()
                        (cons (f x (car l2))
                              (zip ’() (cdr l2))))
                    (if (null? l2)
                        (cons (f (car l1) x)
                              (zip (cdr l1) ’()))
                        (cons (f (car l1) (car l2))
                              (zip (cdr l1) (cdr l2))))))
        (zip l1 l2))
```

```

Answers to 2012a1-test1, page 3

 1.    (D) float -> float -> float

 2.    (C) List.fold_right

 3.    (A) ((3-4)/5)-6

 4.    (A) compose

 5.    (C) A structure on the heap, used to hold variables of an outer
           function when referenced by an inner function.

 6.    (B) $line = <$file>;

 7.    (C) thunk

 8.    (B) int list

 9.    (B) access (static) link

10.    (D) Smalltalk

11.    (B) Edsger Dijkstra

```