```
$Id: 2012q1-soln1,v 1.1 2012-02-13 14:59:58-08 - - $
Answers to 2012a1-test1, page 1

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

_____________________________________________________________________
Question 1. [2]

(define (grep pred? list)
        (if (null? list) '()
            (let ((head (car list))
                  (tail (grep pred? (cdr list))))
                 (if (pred? head) (cons head tail)
                                  tail))))

_____________________________________________________________________
Question 2. [1]

(define (positives list)
        (grep (lambda (x) (> x 0)) list))

_____________________________________________________________________
Question 3. [2]

(define (sum list)
        (define (sum2 list acc)
                (if (null? list) acc
                    (sum2 (cdr list) (+ (car list) acc))))
        (sum2 list 0))

_____________________________________________________________________
Question 4. [1]

(define (sum list) (foldleft 0 (lambda (a b) (+ a b)) list))
(define (sum list) (foldleft 0 + list))

_____________________________________________________________________
Question 5. [2]

Hundreds of answers:
- inclusion (or subtype) polymorphism
     - example shows something object-oriented
- generic (or template) polymorphism
     - example shows someting using Java generics or C++ templates
       or equivalent in another language

_____________________________________________________________________
Question 6. [2]

Hundreds of answers:
- overloading polymorphism
     - example shows a function with same name but different
       signatures (prototypes)
- conversion polymorphism
     - example shows automatic conversion of one type to another,
       e.g. float->int, etc.
```

```
 59:
 60: Answers to 2012a1-test1, page 2
 61:
 62: ____________________________________________________________________
 63: Question 7. [5]
 64:
 65: use strict;
 66: use warnings;
 67: $0 =~ s|.*/||;
 68: my $status = 0;
 69: my %hash;
 70: for my $fname (@ARGV ? @ARGV : "-") {
 71:    open my $file, "<$fname"
 72:         or print STDERR "$0: $fname: $!\n" and $status = 1 and next;
 73:    while (defined (my $line = <$file>)) {
 74:       map {++$hash{$_}} $line =~ m/(\w+)/g;
 75:       ##alternate: map {++$hash{$_}} split m/\W+/, $line;
 76:    }
 77: }
 78: map {print "$_ $hash{$_}\n"} sort keys %hash;
 79: exit $status;
 80:
 81: ____________________________________________________________________
 82: Question 8. [2]
 83:
 84: (define (zip f l1 l2)
 85:     (if (or (null? l1) (null? l2)) '()
 86:         (cons (f (car l1) (car l2)) (zip f (cdr l1) (cdr l2)))
 87: ))
 88:
 89: ____________________________________________________________________
 90: Question 9. [2]
 91:
 92: sub zip {
 93:    my ($f, $l1, $l2) = @_;
 94:    my @a;
 95:    for ($i = 0; $i < @$l1 && $i < @$l2; ++$i) {
 96:       push @a, $f->($l1->[$i], $l2->[$i]);
 97:    }
 98:    return @a
 99: }
100:
101: ____________________________________________________________________
102: Question 10. [1]
103:
104: print "@ARGV\n";
105:
```

```
106:
107: Answers to 2012a1-test1, page 3
108:
109:  1.    (A) Haskell
110:
111:  2.    (A) &&
112:
113:  3.    (D) $ O ( 2 sup n ) $
114:
115:  4.    (A) $ O ( n ) $
116:
117:  5.    (B) (4 5 6)
118:
119:  6.    (B) When the program is linked.
120:
121:  7.    (A) strong and dynamic.
122:
123:  8.    (C) the stack frame of the function in which this function is
124:        nested.
125:
126:  9.    (D) only M, but neither D nor U.
127:
128: 10.    (A) &(&p)
129:
130: 11.    (C) FORTRAN
131:
```