```
$Id: 2014q4-soln1,v 1.3 2014-10-29 13:36:09-07 - - $
Answers to 2014q4-test2, page 1
____________________________________________________________________________
Question 1. [2]

Give 1/2 point for each correct answer, but not more than 2 points
total for the question.  Ignore any "Intercal" answer.
Haskell |Language which uses primarily lazy evaluation, based on the $la
Basic   |Kemeny and Kurtz designed this language included in the IBM PC
AWK     |Scripting language designed by Aho, Weinberger, and Kernighan.
Python  |Van Rossum designed this scripting language named after Monty's
Scheme  |Steele and Sussman designed this functional language with lexic


____________________________________________________________________________
Question 2. [2]

brief answer ...................
$lc++, $cc+= length, $wc+= @{[m/(\S+)/g]} while <>;
print "$lc $wc $cc\n";

verbose answer ...................
while ($line = <>) {
   ++$lc;
   $cc += length $line;
   ++wc while $line =~ s/\S+//;
}
print "$lc $wc $cc\n";

____________________________________________________________________________
Question 3. [3]

Object extend [
   fibonacci: n [
      |array a b c|
      array := Array new: n.
      a := 0.
      b := 1.
      1 to: n do: [:i|
         array at: i put: a.
         c := a + b. a := b. b := c.
      ].
      ^ array
   ]
]

____________________________________________________________________________
Question 4. [4]

(define (merge l1 l2)
   (cond ((null? l1) l2)
         ((null? l2) l1)
         (else (let ((a1 (car l1))
                     (a2 (car l2)))
                    (if (< a1 a2)
                        (cons a1 (merge (cdr l1) l2))
                        (cons a2 (merge l1 (cdr l2)))
)))))
```

```

Answers to 2014q4-test2, page 2
________________________________________________________________________
Question 5. [1]

(define (product list)
        (define (prod list p)
                (if (null? list) p
                    (prod (cdr list) (* (car list) p))))
        (prod list 1))


________________________________________________________________________
Question 6. [2]

(define (foldl fn unit list)
        (if (null? list) unit
            (foldl fn (fn unit (car list)) (cdr list))))

________________________________________________________________________
Question 7. [1]

(define (product list) (foldl * 1 list))

________________________________________________________________________
Question 8. [1]

non lambda version ...................
(define (ad n _) (+ n 1))
(define (len list) (foldleft ad 0 list))
lambda version ...................
(define (lenn list) (foldleft (lambda (n _) (+ n 1)) 0 list))

________________________________________________________________________
Question 9. [2]

product := [:vec |
   |prod|
   prod := 1.
   1 to: vec size do: [:i| prod := prod * (vec at: i)].
   prod
].

________________________________________________________________________
Question 10. [1]

$product = 1;
map {$product *= $_} @array;
________________________________________________________________________
Question 11. [2]

(define (reverse list)
        (define (rev list m)
                (if (null? list) m
                    (rev (cdr list) (cons (car list) m))))
        (rev list '()))

```

```

Answers to 2014q4-test2, page 3

 1.    (C) An unevaluated expression passed into a function which may be
           evaluated by the function if needed.

 2.    (B) (cadr '(1 2 3))

 3.    (C) 5

 4.    (A) ((a b) c: d)

 5.    (B) $x$ is bound and $y$ is free.

 6.    (B) multiple (mixin) inheritance of functions (methods) but not
           fields.

 7.    (A) (+ 2 3)

 8.    (B) by the compiler

 9.    (D) reverse

10.    (B) 2 sqrt

11.    (A) strong and dynamic.

12.    (B) 1958, John McCarthy, MIT.

```