Answer Key for Lisp Presentation

@nebu

November 21, 2022

1 Learn (most of) Lisp

1. Write a Lisp procedure `average`, that takes two arguments and computes their arithmetic mean:
$$\texttt{average}(\texttt{x}, \texttt{y}) = \frac{x+y}{2}$$

```
(define (average x y)
  (/ (+ x y) 2))
```

2. Using `average` and the `square` function we defined earlier, define a function:
$$\texttt{mean-square(x, y)} = \frac{x^2+y^2}{2}$$

```
(define (mean-square x y)
  (average (square x) (square y)))
```

2 Most of the rest of Lisp

Write the absolute value function in Lisp:

$$\texttt{abs}(x) = \begin{cases} -x, & x < 0 \\ 0, & x = 0 \\ x, & x > 0 \end{cases}$$

```
(define (abs x)
  (if (< x 0)
      (- x)
      x))
```

or

```
(define (abs x)
  (cond
    ((< x 0) (- x))
    ((= x 0) 0)
    ((> x 0) x)))
```

Challenge question: Write an iterative implementation of Fibonacci that runs in $O(n)$.

```
(define (iter-fib n a b)
  (if (= n 1)
      b
      (iter-fib (dec n) b (+ a b))))

(define (fib n)
  (iter-fib n 0 1))
```

3 Data Structures from Nothing at All

Here's Lisp code to sum all the elements of a list.

```
(define (sum-list lst)
  (if (null? lst)
      0
      (+ (car lst)
         (sum-list (cdr lst)))))

(sum-list (list 1 2 3 4)) ;; 10
```

Write a higher-order function `fold(fn, init, lst)` that combines all the elements of `lst` and `init` using the binary function `fn`. Then we should be able to do `sum-list` as:

```
(fold + 0 lst)
```

Solution:

```
(define (fold fn init lst)
  (if (null? lst)
      init
      (fn (car lst)
          (fold fn init (cdr lst)))))

(fold + 0 (list 1 2 3 4))
```

```
10
```