2.1

```
(define (make-rat n d)
  (let ((g (gcd n d))
        (sign (if (> d 0) 1 -1)))
    (cons (* sign (/ n g))
          (* sign (/ d g)))))
```

2.2

```
(define (make-segment point1 point2)
  (cons point1 point2))

(define (make-point x y)
  (cons x y))

(define (start-segment segment)
  (car segment))

(define (end-segment segment)
  (cdr segment))

(define (x-point point)
  (car point))

(define (y-point point)
  (cdr point))

(define (print-point p)
  (display "(")
  (display (x-point p))
  (display ",")
  (display (y-point p))
  (display ")"))

(define (midpoint p1 p2)
  (make-point (/ (+ (x-point p1) (x-point p2)) 2)
              (/ (+ (y-point p1) (y-point p2)) 2)))
```

2.3

```
(define (make-rect p1 p2)
  (cons p1 p2))

(define (make-point x y)
  (cons x y))

(define (getfirst rec)
  (car rec))

(define (getsecond rec)
  (cdr rec))

(define (getx point)
  (car point))

(define (gety point)
  (cdr point))

(define (area rect)
  (* (width rect) (height rect)))

(define (perimeter rect)
  (+ (* 2 (width rect)) (* 2 (height rect))))

(define (width r)
  (- (getx (getsecond r)) (getx (getfirst r))))

(define (height r)
  (- (gety (getfirst r)) (gety (getsecond r))))
```

2.4

```
(define (cons1 x y)
  (lambda (m) (m x y)))

(define (car1 z)
  (z (lambda (p q) p)))

(define (cdr1 z)
  (z (lambda (p q) q)))
```

2.7

```
(define (lower-bound x)
  (min (car x) (cdr x)))

(define (upper-bound x)
  (max (car x) (cdr x)))
```

2.8

```
(define (sub-interval x y)
   (make-interval (- (lower-bound x) (upper-bound y))
                  (- (upper-bound x) (lower-bound y))))
```

2.9

I wasn’t sure about this question.