1. `(duple n x)` returns a list containing `n` copies of `x`

```
(define (duple n x)
  (cond
    [(= n 0) empty]
    [else (cons x (duple (- n 1) x))]))

;; Examples:
> (duple 2 3)
(3 3)
> (duple 4 '(ho ho))
((ho ho) (ho ho) (ho ho) (ho ho))
> (duple 0 '(blah))
()
```

2. `(invert L)`, where `L` is a list of 2-lists (lists of length two), returns a list with each 2-list reversed.

```
(define (invert L)
  (cond
    [(empty? L) empty]
    [(cons? L) (cons (list (second (first L))
                           (first (first L)))
                     (invert (rest L)))]))

;; Examples:
> (invert '((a 1) (a 2) (b 1) (b 2)))
((1 a) (2 a) (1 b) (2 b))
```

3. `(list-set L n x)` returns a list like `L`, except that the `n`-th element, using zero-based indexing, is `x`.

```
(define (list-set L n x)
  (cond
    [(= n 0) (cons x (rest L))]
    [else (cons (first L) (list-set (rest L) (- n 1) x))]))

;; Examples:
> (list-set '(a b c d) 2 '(1 2))
(a b (1 2) d)
```

4. `(down L)` wraps parentheses around each top-level element of `L`.

```
(define (down L)
  (cond
    [(empty? L) empty]
    [(cons? L) (cons (list (first L)) (down (rest L)))]))

;; Examples:
> (down '(1 2 3))
((1) (2) (3))
> (down '((a) (fine) (idea)))
(((a)) ((fine)) ((idea)))
> (down '(a (more (complicated)) object))
((a) ((more (complicated))) (object))
```

5. `(up L)` removes a pair of parentheses from each top-level element of `L`. If a top-level element is not a list, it is included in the result, as is.

```
(define (up L)
  (cond
    [(empty? L) empty]
    [(cons? L) (cond
                 [(list? (first L)) (append (first L)
                                            (up (rest L)))]
                 [else (cons (first L)
                             (up (rest L)))])]))

;; Examples:
> (up '((1 2) (3 4)))
(1 2 3 4)
> (up '((x (y)) z))
(x (y) z)
```

6. `(swapper s1 s2 L)` returns a list the same as `L`, but with all occurrences of `s1` replaced by `s2` and all occurrences of `s2` replaced by `s1`.

```
(define (swapper s1 s2 L)
  (cond
    [(empty? L) empty]
    [(cons? L) (cons (cond
                       [(list? (first L)) (swapper s1 s2 (first L))]
                       [(symbol=? (first L) s1) s2]
                       [(symbol=? (first L) s2) s1]
                       [else (first L)])
                     (swapper s1 s2 (rest L)))]))

;; Examples:
> (swapper 'a 'd '(a b c d))
(d b d a)
> (swapper 'a 'd '(a d () c d))
(d a () c a)
> (swapper 'x 'y '((x) y (z (x))))
((y) x (z (y)))
```

7. `(flatten L)` returns a list of the symbols contained in `L` in the order in which they occur when `L` is printed. Intuitively, `flatten` removes all the inner parentheses from its argument.

```
(define (flatten L)
  (cond
    [(empty? L) empty]
    [(cons? L) (append (cond
                         [(list? (first L)) (flatten (first L))]
                         [else (list (first L))])
                       (flatten (rest L)))]))

;; Examples:
> (flatten '(a b c))
(a b c)
> (flatten '((a) () (b ()) () (c)))
(a b c)
> (flatten '((a b) c (((d)) e)))
(a b c d e)
> (flatten '(a b (() (c))))
(a b c)
```

8. `(merge L M)`, where `L` and `M` are lists of numbers that are sorted in ascending order, returns a sorted list of all the numbers in `L` and `M`.

```
(define (merge L M)
  (cond
    [(empty? L) M]
    [(empty? M) L]
    [(< (first L) (first M)) (cons (first L) (merge (rest L) M))]
    [else (cons (first M) (merge L (rest M)))]))

;; Examples:
> (merge '(1 4) '(1 2 8))
(1 1 2 4 8)
> (merge '(35 62 81 90 91) '(3 83 85 90))
(3 35 62 81 83 85 90 90 91)
```

9. `(sort L)` returns a list of the elements of `L` in increasing order.

```
(define (before L n)
  (cond
    [(= n 0) empty]
    [else (cons (first L) (before (rest L) (- n 1)))]))

(define (after L n)
  (cond
    [(= n 0) L]
    [else (after (rest L) (- n 1))]))

(define (mid-length L) (truncate (/ (length L) 2)))

(define (sort L)
  (cond
    [(empty? L) empty]
    [(cons? L) (cond
                 [(empty? (rest L)) (list (first L))]
                 [else (merge (sort (before L (mid-length L)))
                              (sort (after L (mid-length L))))])]))

;; Examples:
> (sort '(8 2 5 2 3))
(2 2 3 5 8)
```