SOLUCIONES:

Ejercicio 1:

```
(define (diferencias-mayor-que lista n)
   (cond
      ((empty? lista) 0)
      ((empty? (cdr lista)) 0)
      ((> (abs (- (car lista) (cadr lista))) n)
           (+ 1 (diferencias-mayor-que (cdr lista) n)))
      (else
           (diferencias-mayor-que (cdr lista) n))))
```

Ejercicio 2:

```
(define (anyade-0 lista)
  (begin
    (set-cdr! (list-tail lista (- (length lista) 1))
           (list 0))
    lista))

(define (inc-nth lista n)
   (if (> n (- (length lista) 1))
      (inc-nth (anyade-0 lista) n)
      (let ((pos-n (list-tail lista n)))
        (set-car! pos-n (+ (car pos-n) 1)))))

(define (histograma lista)
  (let ((lista-aux '(0)))
    (histograma-aux lista lista-aux)
    lista-aux))

(define (histograma-aux lista list-aux)
  (if (null? lista)
      list-aux
      (begin
        (inc-nth list-aux (car lista))
        (histograma-aux (cdr lista) list-aux))))
```

Ejercicio 3:

```scheme
(define (sort! lst)
  (if (null? lst) '()
      (insert! lst (sort! (cdr lst)))))

(define (insert! value-pair sorted)
  (cond
    ((null? sorted)
        (set-cdr! value-pair '())
        value-pair)
    ((< (car value-pair) (car sorted))
        (set-cdr! value-pair sorted)
        value-pair)
    (else
        (set-cdr! sorted (insert! value-pair (cdr sorted)))
        sorted)))
```

Ejercicio 4:

a)

```scheme
(define (lista-binaria-a-num lista)
  (if (null? lista) 0
      (+ (lista-binaria-a-num (cdr lista))
         (* (car lista) (aux (- (length lista) 1))))))

(define (aux n)
  (if (= n 0) 1
      (* 2 (aux (- n 1)))))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(define (num-a-lista-bin num)
  (cond
    ((= num 1) (list num))
    (else (append (num-a-lista-bin (parte-entera num 2)) (list (remainder num 2))))))

(define (parte-entera n m)
      (let ((resto (remainder n m)))
        (/ (- n resto) m)))
```

b)

```scheme
(define (reduce-tree tree)
  (make-tree (lista-binaria-a-num (dato tree))
             (map reduce-tree (hijos tree)) ))

(define (amplia-tree tree)
  (make-tree (num-a-lista-bin (dato tree))
             (map amplia-tree (hijos tree)) ))
```

Ejercicio 5

Devuelve 13