1.

a. (define find\_help (lambda (n lst) (if (null? lst) 0 (if (= (car lst) n) 1 (find\_help n (cdr lst))))))

(define find (lambda (n lst) (if (null? lst) (list) (if (= (find\_help n (car lst)) 1) (find n (cdr lst)) (cons (car lst) (find n (cdr lst)))))))

b. (define appendtwostring (lambda (lst1 lst2) (if (null? lst1) lst2 (if (null? lst2) lst1 (cons (car lst1) (appendtwostring (cdr lst1) lst2))))))

(define append (lambda (lst) (if (null? (cdr lst)) (car lst) (appendtwostring (car lst) (append (cdr lst))))))

(define max\_help (lambda (lst max) (if (null? lst) max (if (> (car lst) max) (max\_help (cdr lst) (car lst)) (max\_help (cdr lst) max)))))

(define max (lambda (lst) (if (null? lst) 0 (max\_help lst (car lst)))))

(define delete\_help (lambda (n lst) (if (null? lst) lst (if (= n (car lst)) (cdr lst) (cons (car lst) (delete\_help n (cdr lst)))))))

(define findx (lambda (n lst) (if (null? lst) (list) (if (= (find\_help n (car lst)) 1) (cons (delete\_help n (car lst)) (cdr lst)) (cons (delete\_help n (car lst)) (findx n (cdr lst)))))))

(define sort (lambda (lst) (if (null? (cdr (car lst))) (car (car lst)) (cons (sort (findx (max (append lst)) lst)) (max (append lst))))))

2. (define triangle\_help (lambda (n) (if (= 0 n) (list) (cons 0 (triangle\_help (- n 1))))))

(define triangle (lambda (n) (if (= 0 n) (list) (cons (triangle\_help n) (triangle (-n 1))))))