**17.1**

> (car '(Rod Chris Colin Hugh Paul)) **Rod**

> (cadr '(Rod Chris Colin Hugh Paul)) **Chris**

> (cdr '(Rod Chris Colin Hugh Paul)) **(Chris Colin Hugh Paul)**

> (car 'Rod) **error**

> (cons '(Rod Argent) '(Chris White)) **((Rod Argent) Chris White)**

> (append '(Rod Argent) '(Chris White)) **(rod argent chris white)**

> (list '(Rod Argent) '(Chris White)) **((rod argent) (chris white))**

> (caadr '((Rod Argent) (Chris White)

(Colin Blunstone) (Hugh Grundy) (Paul Atkinson)))

**'Chris**

> (assoc 'Colin '((Rod Argent) (Chris White)

(Colin Blunstone) (Hugh Grundy) (Paul Atkinson)))

**'(Colin Blunstone)**

> (assoc 'Argent '((Rod Argent) (Chris White)

(Colin Blunstone) (Hugh Grundy) (Paul Atkinson)))

**#f**

> (assoc ‘Chris ‘(Chris Colin Hugh Paul))

**Error**

**17.2**

;;; Part 1

(define (f1 se1 se2)

(list (list (cadr se1) (caddr se1) (car se2))))

;;; Part 2

(define (f2 se1 se2)

(list (cdr se1) (cadr se2)))

;;; Part 3

(define (f3 se1 se2)

(append se1 se1))

;;; Part 4

(define (f4 se1 se2)

(list (list (car se1) (car se2)) (append (cdr se1) (cdr se2))))

**17.8**

(define (membr x lst)

(if (null? lst)

#f

(if (equal? x (car lst)) #t (membr x (cdr lst)))))

**17.9**

(define (list-ref lst num)

(cond ((null? lst) #f)

((equal? num 0) (car lst))

(else (list-ref (cdr lst) (- num 1)))))

**17.10**

(define (length lst)

(if (null? lst)

0

(+ 1 (length (cdr lst)))))

**17.11**

(define (before-in-list? lst wd1 wd2)

(cond ((null? lst) #f)

((equal? (car lst) wd1) (member? wd2 lst))

(else (before-in-list? (cdr lst) wd1 wd2))))

**17.12**

(define (flatten lst)

(if (null? lst)

'()

(if (word? lst)

lst

(reduce se (map (lambda (x) (flatten x)) lst)))))

**17.14**

(define (branch numlst lst)

(cond ((null? numlst) lst)

((> (car numlst) (length lst)) #f)

(else (branch (cdr numlst) (branch-helper (car numlst) lst)))))

(define (branch-helper num lst)

(if (equal? num 1)

(car lst)

(branch-helper (- num 1) (cdr lst))))