Part 1: (23 points)

Question 1. 4 points

procedure: number, A→ number

Question 2. 3 points

error, not a procedure

Question 3. 3 points

18, number

Question 4. 3 points

(4 3)

Question 5. 3 points

16, number

Question 6. 4 points

procedure: number → number

Question 7. 3 points

([proc] 2 3)
Part 2: (18 points)

Question 8.
5 points

(define (add-em-up lst)
  (if (null? lst)
      0
      (+ (registered (car lst))
          (add-em-up (cdr lst)))))

5 points

(define (add-em-up lst)
  (define (aux  sum  todo)
    (if (null? todo)
        num
        (aux (+ sum (registered (car todo)))
             (cdr todo)))
  (aux 0 lst))

Question 9.
8 points

(define (add-em-up lst)
  (if (null? lst)
      0
      (+ (registered (car lst))
          (add-em-up (cdr lst)))))
Part 3: (24 points)

Question 10.
6 points

(define (helper tag stats)
  (if (null? stats)
      '()
      (cons (list (list tag (term (car stats)))
                (registered (car stats)))
            (helper tag (cdr stats)))))

Question 11.
6 points

(define (convert-all data)
  (if (no-classes? data)
      '()
      (APPEND (CONVERT-CLASS (NEXT-CLASS DATA))
               (CONVERT-ALL (REST-CLASSES DATA))))

Question 12.
6 points

(define (make-class-extractor what-class)
  (lambda (x) (= what-class (caar x))))

Question 13.
6 points

(define (make-class-extractor what-class what-term)
  (lambda (x) (equal? (list what-class what-term) (car x))))
Part 4: (15 points)

Question 14.  3 points
linear B

Question 15.  3 points
constant A

Question 16.  5 points
quadratic D

Question 17.  4 points
linear B
Part 5: (20 points)

Question 17.  4 points
Both option A and B will work as described.

Question 19.  8 points
(define (mul a b)
    ((REPEAT (LAMBDA (X) (+ A X)) B) 0))

Question 20.  8 points
(define (my-exp a b)
    ((REPEAT (LAMBDA (X) (* A X)) B) 1))