1. Convert the following code to continuation-passing style; i.e., all calls must be tail-recursive.

   (define append (el li)
     (if (null? li)
         (list el)
         (cons (car li) (append el (cdr li)))))

2. Convert the following code to CPS, using the CPS version of `append` above.

   (define reverse (li)
     (if (null? li) li (append (car li) (reverse (cdr li)))))

3. Reconstruct the type of the following expression and all its subexpressions, or justify why the type cannot be reconstructed within the \( R \) framework we have developed to support parametric polymorphism:
(let ((id (lambda (x) x)))
  (id id 1))

4. Show important steps during the reconstruction of the type of variables and subexpressions of the following expression, and write down the type of the overall expression.

(rec fib (proc (n)
  (if (< 2 n)
    1
    (+ (call fib (- n 1)) (call fib (- n 2))))))

Total: 10