1

a Endrekursive Prozeduren sind Prozeduren deren Definitionen ausschließlich kontextfreie rekursive Aufrufe enthalten. Der Berechnungsprozess ist iterativ, da seine Größe konstant bleibt.

B Promise kapselt einen Ausdruck und schottet ihn ab um ihn bei Bedarf später via Force auswerten zu können.

C Parametrisch polymorphe Prozeduren sind Prozeduren deren Signaturen Signaturvariablen enthält und daher lediglich reproduzieren, verwerfen oder weitergeben können.

D Currying - Umwandlung mehrstelliger Abstraktionen in eine Komposition einstelliger Abstraktionen, die Argumente sukzessive aufnimmt

4

a (: fold (%a (%b %a -> %a) (list -of %b) -> %a))

(: append ((list -of %a) (list -of %a) -> (list -of %a)))

5

a (: mystery ((%a → %a) %a (%a → boolean) → (list-of %a)))

b (mystery (lambda (x) (+ x 1))
1
(lambda (x) (= x 10)))

6.

```
(: and-filter ((%a -> boolean) (%a -> %b) (list-of %a) -> (list-of %b)))
(check-expect (and-filter (lambda (x) (even? x)) (lambda (x) (+ x 10))
 (list 1 2 5 11 18 19 20))  (list 12 28 30))


(define and-filter
  (lambda (p? f xs)
    (cond ((empty? xs) empty)
          ((pair? xs) (cond ((p? (first xs))
                             (make-pair (f (first xs))
                                        (and-filter p? f (rest xs))))

                            (else (and-filter p? f (rest xs))))))))
```

7.

b)

```
; Ein Stream besteht aus
; - einem ersten Element (head)
; - einem Promise, den Rest des Streams generieren zu können (tail)
(define stream-of
  (lambda (t)
    (signature (cons-of t (promise (stream-of t))))))
```

c)

```
(: stream-take (natural (stream-of %a) -> (list-of %a)))
(define stream-take
  (lambda (n str)
    (if (= n 0)
        empty
        (make-pair (head str)
                   (stream-take (- n 1) (force (tail str)))))))
```

d)

```
; Beispiel:
; Stream mit Zahlen ab n erzeugen
(: to (number -> (stream-of number)))
(define to
  (lambda (n)
    (make-cons n (lambda () (to (+ n 1))))))
```