1 – A) O valor de (f2 5) será 25 porque na linha onde está “ **(define f2 f1)** ” ele está sendo definido assumindo a definição de f1, que é usado para calcular um número ao quadrado. Assim chamando f2 seria a mesma coisa de que se estivesse sendo chamado a definição até o momento de f1.

1 – B) Agora o valor de (f1 5) ainda será 25, mas o valor de (f2 5) agora será 15, pois o f2 agora passou a assumir uma função que retorna o “valor + 10” enquanto f1 não sofreu nenhuma modificação diretamente.

**Definição:**

(define (valorSacoMoedas\_aux sacoMoedas soma)

(if (null? sacoMoedas)

soma

(valorSacoMoedas\_aux (cdr sacoMoedas) (if (eq? (car sacoMoedas) 1)

(+ soma 100)

(+ soma (car sacoMoedas))))))

(define (valorSacoMoedas sacoMoedas)

(/ (valorSacoMoedas\_aux sacoMoedas 0) 100))

**Solução dado “(valorSacoMoedas (list 1 10 20))” (teste de mesa):**

valorSacoMoedas => sacoMoedas = (list 1 10 20)

valorSacoMoedas\_aux => sacoMoedas = (list 1 10 20), soma = 0

(if (null? sacoMoedas) => #f

(if (eq? (car sacoMoedas) 1) => #t

(+ soma 100) => soma = 100

valorSacoMoedas\_aux => sacoMoedas = (list 10 20), soma = 100

(if (null? sacoMoedas) => #f

(if (eq? (car sacoMoedas) 1) => #f

(+ soma (car sacoMoedas)) => soma = 110

valorSacoMoedas\_aux => sacoMoedas = (list 20), soma = 110

(if (null? sacoMoedas) => #f

(if (eq? (car sacoMoedas) 1) => #f

(+ soma (car sacoMoedas)) => soma = 130

valorSacoMoedas\_aux => sacoMoedas = (list), soma = 130

(if (null? sacoMoedas) => #t

soma => 130

(/ (valorSacoMoedas\_aux sacoMoedas 0) 100) => 1.3

retorna => 1.3

**Definição:**

(define (listasIguais lista1 lista2)

(if (and (null? lista1) (null? lista2))

"Verdadeiro"

(if (or (and (null? lista1) (not (null? lista2))) (and (null? lista2) (not (null? lista1))))

"Falso"

(if (eq? (car lista1) (car lista2))

(listasIguais (cdr lista1) (cdr lista2))

"Falso"))))

**Solução dado “(listasIguais (list 1 2 3) (list 1 2))” (teste de mesa):**

listasIguais => lista1 = (list 1 2 3), lista2 = (list 1 2)

(if (and (null? lista1) (null? lista2)) => #f

(if (or (and (null? lista1) (not (null? lista2))) (and (null? lista2) (not (null? lista1)))) => #f

(if (eq? (car lista1) (car lista2)) => #t

listasIguais => lista1 = (list 2 3), lista2 = (list 2)

(if (and (null? lista1) (null? lista2)) => #f

(if (or (and (null? lista1) (not (null? lista2))) (and (null? lista2) (not (null? lista1)))) => #f

(if (eq? (car lista1) (car lista2)) => #t

listasIguais => lista1 = (list 3), lista2 = (list)

(if (and (null? lista1) (null? lista2)) => #f

(if (or (and (null? lista1) (not (null? lista2))) (and (null? lista2) (not (null? lista1)))) => #t

Retorna => “Falso”

(define (contem n lista)

(if (null? lista)

#f

(if (eq? n (car lista))

#t

(contem n (cdr lista)))))

(define (somatorio\_aux n n\_aux)

(if (zero? n)

n\_aux

(somatorio\_aux (- n 1) (+ n\_aux n))))

(define (somatorio n)

(somatorio\_aux n 0))

(define (len\_aux lista n)

(if (null? lista)

n

(len\_aux (cdr lista) (+ n 1))))

(define (len lista)

(len\_aux lista 0))