01. [Sebesta, 2000] Defina forma funcional e transparência referencial?

02. [Sebesta, 2000] Quais tipos de dados faziam parte do LISP original?

03. [Sebesta, 2000] Qual é a diferença entre `EQ?`, `EQV?` e =?

04. [Sebesta, 2000] Quais são as diferenças entre o método de avaliação usado para a forma especial da Scheme, `DEFINE`, e a usada para suas funções primitivas?

05. [Sebesta, 2000] Quais são as duas formas de `DEFINE`?

06. [Sebesta, 2000] Descreva a semântica de `COND`.

07. [Sebesta, 2000] Descreva a semântica de `LET`.

08. [Sebesta, 2000] Por que recursos imperativos foram adicionados à maioria dos dialetos do LISP?

09. [Sebesta, 2000] Sob quais aspectos o COMMON LISP e a Scheme são opostos?

10. [Sebesta, 2000] Qual regra de escopo é usada na Scheme? No COMMON LISP? Na ML? Na Haskell?

11. [Sebesta, 2000] Quais são as três características que deixam a ML muito diferente da Scheme?

12. [Sebesta, 2000] O que é inferência de tipo, quando usada na ML?

13. [Sebesta, 2000] Quais são os três recursos da Haskell que a tornam muito diferente da Scheme?

14. [Sebesta, 2000] O que significa avaliação preguiçosa?

15. [Sebesta, 2000] Escreva uma função Scheme que retorne o inverso de seu parâmetro de lista simples.

16. [Sebesta, 2000] Escreva uma função de predicado em Scheme que teste a igualdade de estrutura de duas listas dadas. Ambas são estruturalmente iguais se tiverem a mesma estrutura de lista, não obstante seus átomos poderem ser diferentes.

17. [Sebesta, 2000] Escreva uma função Scheme que retorne a união de dois parâmetros de lista simples que representam conjuntos.

18. [Sebesta, 2000] Escreva uma função Scheme com dois parâmetros, um átomo e uma lista, que retorne a lista com todas as ocorrências, não importa quão profundas, do átomo dado excluído. A lista retornada não pode conter nada em lugar dos átomos excluídos.

19. [Sebesta, 2000] Escreva uma função Scheme que tome uma lista como parâmetro e retorne-a com o segundo elemento de nível máximo removido. Se a lista dada não tiver dois elementos, a função deve retornar ( ).

20. [Sebesta, 2000] Leia o documento de Jonh Backus sobre a FP (Backus, 1978) e compare os recursos da Scheme discutidos neste capítulo com os recursos correspondentes da FP.

21. [Sebesta, 2000] Encontre definições das funções Scheme `EVAL` e `APPLY`, e explique suas ações.

22. [Sebesta, 2000] Um dos mais modernos e completos ambientes de programação para qualquer linguagem é o sistema INTERLISP para LISP, conforme é descrito em *The INTERLISP Programming Environment*, de Teitelman e Masinter (IEEE Computer, Vol. 14, No. 4, abril de 1981). Leia esse artigo cuidadosamente e compare a dificuldade de escrever programas LISP em seu sistema com aquele que usa o INTERLISP (supondo que você normalmente não use o INTERLISP).

23. [Sebesta, 2000] Consulte um livro sobre programação LISP e determine quais argumentos sustentam a inclusão do recurso `PROG` no LISP.

24. [Sebesta, 2000] Uma linguagem funcional poderia usar alguma estrutura de dados que não seja uma lista. Por exemplo, ela poderia usar sequências de símbolos. Quais primitivas essa linguagem teria em lugar das primitivas CAR, CDR e CONS da Scheme?

25. [Sebesta, 2000] O que a seguinte função Scheme faz?

```
(define (y s lis)
  (cond
    ((null? Lis) ' ())
    ((equal? s (car lis)) lis)
    (else (y s (cdr lis)))
))
```

26. [Sebesta, 2000] O que a seguinte função Scheme faz?

```
(define (x lis)
  (cond
    ((null? Lis) 0)
    ((not (list? (car lis)))
      (cond
        ((eq? (car lis) nil)(x (cdr lis)))
        (else (+ 1 (x (cdr lis))))))
    (else (+ (x (car lis))(x (cdr lis))))
))
```

27. Apresentar graficamente as seguintes listas em LISP.

a) `(a b (c d) e f)`

b) `((x y)((z w) t) k)`

28. Converter as expressões aritméticas a seguir para LISP.

a) 5 + 7 * 4 – 6

b) 5 * (9 – 7 * 4) – (4 / 5 + 3)

29. Apresente as expressões em LISP para retirar das listas a letra 'x'.

a) `((a b c)(x d e))`

b) `((a (b c) x) e f)`

30. Desenvolver uma função em LISP que retorne o calculo do imposto devido, com base na renda da pessoa, utilizando a tabela a seguir.

Renda	Imposto
Até R$ 5.000,00	5,00%
Entre R$ 5.000,00 e R$ 10.000,00	7,00%
Acima de R$ 10.000,00	10,00%

31. Apresente a expressão em LISP, utilizando exclusivamente os operadores CAR e CDR, para retirar o elemento X da lista `(A B (C (D E)(F X) G) H)`.

32. Desenvolva uma função ou um conjunto de funções em LISP que calcule o valor de e^x utilizando a fórmula

$$e^x = x^0/0! + x^1/1! + x^2/2! + x^3/3! + ... + x^n/n!$$

O valor de n será fornecido pelo usuário, devendo ser um valor inteiro e positivo.

O valor de x será fornecido pelo usuário, podendo ser um valor (inteiro ou real) qualquer.

Por exemplo, caso o valor fornecido pelo usuário para n seja 4 e para x seja 2, o programa deverá apresentar como resposta o valor 7, ou seja, $2^0/0! + 2^1/1! + 2^2/2! + 2^3/3! + 2^4/4!$.

Caso o usuário forneça um valor inválido para n, o programa deverá apresentar como resposta o valor `nil`.

33. Desenvolva uma função ou um conjunto de funções em LISP que apresente o valor aproximado da raiz quadrada de um número A, por meio de n iterações, através da sequência de aproximação $x_n = (x_{n-1} + A/x_{n-1})/2$, com $x_1 = 1$ e $n \in \aleph$.

O número de iterações e o valor de A serão fornecidos pelo usuário, devendo ser um valor inteiro e positivo.

Por exemplo, caso o valor fornecido pelo usuário para o número de iterações seja 5 e para A seja 3, o programa deverá apresentar como resposta o valor `1.732050810`, obtido pela sequência de valores

$x_1 = 1$
$x_2 = (x_1 + 3/x_1) / 2 = 2$
$x_3 = (x_2 + 3/x_2) / 2 = 1.75$
$x_4 = (x_3 + 3/x_3) / 2 = 1.732142857$
$x_5 = (x_4 + 3/x_4) / 2 = 1.732050810$

Caso o usuário forneça um valor inválido para o número de iterações ou para A, o programa deverá presentar como resposta o valor `nil`.

34. Desenvolva uma função ou um conjunto de funções em LISP que calcule o valor de π com a série infinita

$$\pi = 4 - 4/3 + 4/5 - 4/7 + 4/9 - 4/11 + ...$$

O número de termos será fornecido pelo usuário, devendo ser um valor inteiro e positivo.

Por exemplo, caso o número de termos fornecido pelo usuário seja 5, o programa deverá apresentar como resposta o valor `3.34`, ou seja, `4 - 4/3 + 4/5 - 4/7 + 4/9`.

Caso o usuário forneça um valor inválido para o número de termos, o programa deverá presentar como resposta o valor `nil`.

35. Desenvolva uma função ou um conjunto de funções em LISP que dado dois números inteiros positivos, determine quantas vezes o primeiro divide exatamente o segundo. Se o primeiro número não divide o segundo, o número de vezes é zero. Os valores dos dois números serão fornecidos pelo usuário, devendo ser valores inteiros e positivos.

Por exemplo, caso os valores fornecidos pelo usuário sejam 2 e 8, o programa deverá apresentar como resposta o valor 3, ou seja, `8 / 2 = 4 / 2 = 2 / 2 = 1`.

Caso o usuário forneça um valor inválido para alguns dos dois números, o programa deverá presentar como resposta o valor `nil`.

36. Desenvolver uma função ou um conjunto de funções em LISP, que apresente o valor de g utilizando a série

$$g = 1/1! - 2/1! + 3/2! - 4/3! + 5/5! - 6/8! + 7/13! - ... + n/F_n!.$$

Os termos da sequência de Fibonacci, são definidos recursivamente pela fórmula $F_n = F_{n-1} + F_{n-2}$, sendo $F_1 = 1$ e $F_2 = 1$.

O valor de n será fornecido pelo usuário, devendo ser um valor inteiro maior do que zero.

Por exemplo, caso o valor fornecido pelo usuário para n seja 5, o programa deverá apresentar como resposta o valor -1/8, ou seja, 1/1! - 2/1! + 3/2! - 4/3! + 5/5!.

Caso o usuário forneça um valor inválido para n, o programa deverá apresentar como resposta o valor nil.

37. Desenvolva uma função ou um conjunto de funções em LISP que apresente o produto dos termos da série de Fibonacci. A série de Fibonacci é formada pela sequência

1, 1, 2, 3, 5, 8, 13, 21, 34, ...

A série de Fibonacci é de grande importância matemática, e a lei básica é que a partir do terceiro termo, todos os termos são a soma dos dois últimos.

O número de termos será fornecido pelo usuário, devendo ser um valor inteiro e positivo.

Por exemplo, caso o número de termos fornecido pelo usuário seja 7, o programa deverá apresentar como resposta o valor 3120, ou seja, 1 * 1 * 2 * 3 * 5 * 8 * 13.

Caso o usuário forneça um valor inválido para o número de termos, o programa deverá presentar como resposta o valor nil.

38. Desenvolva uma função ou um conjunto de funções em LISP que calcule o valor da série infinita

$$H = 1^1/1! - 2^2/2! + 3^3/3! - 4^4/4! + ...$$

O número de termos será fornecido pelo usuário, devendo ser um valor inteiro e positivo.

Por exemplo, caso o número de termos fornecido pelo usuário seja 5, o programa deverá apresentar como resposta o valor 18.88, ou seja, $1^1/1! - 2^2/2! + 3^3/3! - 4^4/4! + 5^5/5!$.

Caso o usuário forneça um valor inválido para o número de termos, o programa deverá presentar como resposta o valor nil.