Autorreferência e recursividade - Prática

Marco A L Barbosa malbarbo.pro.br

1. Projete uma função que concatene todas os elementos de uma lista de strings.
2. Projete uma função que calcule o produto de todos os elementos de uma lista de números. Dica: considere que o produto dos elementos de uma lista vazia é 1.
3. Projete uma função que determine a quantidade de elementos de uma lista de números. Dica: o primeiro elemento pode ser ignorado.
4. Projete uma função que crie uma lista de números a partir de uma lista de strings convertendo cada string para um número. Assuma que todas as strings representam números válidos. Use a função `string->number` para converter uma string para um número, como em `(string->number "123") ; resulta em 123`.
5. Projete uma função que remova todas as strings que não começam com `A` de uma lista de strings.
6. Projete uma função que crie uma nova lista removendo todas os valores nulos de uma lista de inteiros.
7. Projete uma função que verifique se todos os elementos de uma lista de booleanos são verdadeiro. Se a sua implementação inicial utilizar `if` para fazer a chamada recursiva, simplifique o código removendo o `if`.
8. Projete uma função que verifique se algum dos elementos de uma lista de booleanos é verdadeiro. Se a sua implementação inicial utilizar `if` para fazer a chamada recursiva, simplifique o código removendo o `if`.
9. Projete uma função que devolva o último elemento de uma lista não vazia de números.
10. Projete uma função que determine o tamanho máximo entre todas as strings de uma lista não vazia de strings.
11. Projete uma função que verifique se uma lista de números está em ordem não decrescente. Dica: use dois casos base.
12. Defina uma função que que receba como entrada uma lista `lst` e devolva uma lista com os mesmos elementos de `lst` mas em ordem contrária.
13. Projete uma função que indique se em uma lista de inteiros existem mais valores positivos ou negativos. Dica: crie um plano e use funções auxiliares.
14. Projete uma função que calcule a amplitude dos valores de uma lista não vazia de números, isto é, a diferença entre o valor máximo e mínimo da lista. Dica: crie um plano e use funções auxiliares.
15. Projete uma função que receba como parâmetro um número natural n e calcule o produto dos números $1, 2, \cdots, n$.
16. Projete uma função que receba como entrada um número a (diferente de 0) e um número natural n e calcule o valor a^n.
17. Projete uma função que receba como entrada dois números naturais maiores que zero, n e x, e devolva uma lista com os divisores de x que são menores ou iguais a n (não se preocupe com a ordem dos valores na resposta).

18. Um número natural é perfeito se a soma dos seu divisores, exceto ele mesmo, é igual a ele. Por exemplo, o número 6 é perfeito porque $6 = 1 + 2 + 3$. Projete uma função que use uma sequência de etapas para verificar se um número natural é perfeito.

19. Recursão indireta é quando duas ou mais funções chamam uma a outra. Defina duas funções `impar?` e `par?`, uma em termos da outra.

20. (Desafio) Utilizando apenas as funções primitivas `zero?`, `add1` e `sub1`, escreva as funções `+`, `-` e `*`. Cada função deve receber como parâmetro dois números naturais e executar a operação aritmética apropriada.

21. (Desafio) Dados duas listas `lsta` e `lstb`, defina uma função que verifique se `lsta` é prefixo de `lstb`, isto é `lstb` começa com `lsta`.