

INSTRUÇÕES: LEIA COM ATENÇÃO !!!

- i. A prova deve ser respondida usando o link do JSFiddle fornecido pelo professor e, ao fim da prova, a dupla deve realizar upload do link atualizado do JSFiddle usando o Classroom; o nome completo dos dois integrantes da dupla deve constar como comentário na primeira linha do arquivo de solução no JSFiddle.
- ii. Para todas as questões, o nome da função principal e seus parâmetros devem ser respeitados/mantidos pois existem casos de teste que fazem uso dessa nomenclatura ao final do arquivo. Existem duas exceções permissivas: (i) alterar a representação do parâmetro de listas para seu formato explícito de representação, `[x,...xs]`, por exemplo, e (ii) acrescentar um parâmetro inicializado, já que isso não altera as chamadas de funções já definidas nos casos de teste.
- iii. Esses casos de teste servem para verificar automaticamente a corretude das soluções, bem como testar se a solução faz uso de recursividade em lista. ⚠ A RECURSIVIDADE EM LISTA é item OBRIGATÓRIO dessa prova e sua ausência ZERA a solução da questão.
- iv. O template usa o `utils.js`, usado no material de aula, portanto, pode-se fazer uso de suas funções, tais como `indef`, `equals`, etc, caso desejado.
- v. Caso sua lógica para solução de alguma questão necessite do conceito de *função de interface*, você deve usar a nomenclatura `helper` para a função auxiliar interna, que irá abrigar a recursividade; o teste de recursividade só identifica a recursividade desses casos de função auxiliar se o nome for `helper`.

PONTUAÇÃO DA PROVA

Cada questão será avaliada por 5 casos de testes diferentes. Cada caso de teste correto vale 0,5 *pts*. Assim, cada questão vale 2,5 *pts*.

QUESTÕES

1. A função `restoSeq(lista,n)` elimina os primeiros `n` elementos da `lista`.

Exemplos:

```
restoSeq([1,2,3,4,5],3) ----> [4,5]
```

```
restoSeq([],2) ----> []
```

2. A função `checaF(f)(lista)` verifica se os valores numéricos de uma `lista` respeitam o seguinte princípio: cada três valores consecutivos respeitam a operação definida pela função `f`.

Exemplos:

```
checaF((x,y,z)=>x+y===z)([1,1,2,3,5,8,13,21]) ----> true
```

```
checaF((x,y,z)=>x<y && y<z)([1,3,5,8,-13,21]) ----> false
```

```
checaF((x,y,z)=>x+y===z)([1,2]) ----> true
```

3. A função `reduceL(acc)(lista)` realiza a operação `reduce` (com operação de soma), só que mantém o somatório parcial em cada posição da lista final. Ou seja, retorna uma nova lista onde cada posição é o somatório acumulado até então.

Exemplos:

`reduceL(0)([1,2,3,4])` ----> `[0,1,3,6,10]` (ou seja: `[0,0+1,0+1+2,0+1+2+3,0+1+2+3+4]`)

`reduceL('a')(['e','i','o','u'])` ----> `['a','ae','aei','aeio','aeiou']`

4. A função `bin2dec(str)` recebe um número natural em formato binário e retorna sua conversão para decimal. A figura abaixo ajuda a ilustrar como converter de binário para decimal.

$$\begin{array}{ccccccc} & & 1 & 1 & 0 & 1 & 0 & 1 & _2 \\ & \swarrow & \swarrow & \swarrow & \swarrow & \swarrow & \swarrow & \swarrow & \\ 1 \times 2^5 & + & 1 \times 2^4 & + & 0 \times 2^3 & + & 1 \times 2^2 & + & 0 \times 2^1 & + & 1 \times 2^0 \\ \downarrow & & \downarrow & & \downarrow & & \downarrow & & \downarrow & & \downarrow \\ 32 & + & 16 & + & 0 & + & 4 & + & 0 & + & 1 = 53 \end{array}$$
$$110101_2 = 53_{10}$$

Exemplos:

`bin2dec("10")` ----> 2

`bin2dec("11010")` ----> 26