Composição de funções

É possível utilizar *redução* para prover um mecanismo de **composição** de funções. A composição de funções permite concatenar sucessivas definições de funções a fim de gerar uma única função representativa desse "pipeline". A composição promove organização de codificação e ajuda na promoção da *legibilidade* da solução de sub-problemas.

Função de composição pode ser definida seguindo o template a seguir.

```
const composicao =
    (...fns) =>
        (valor) => fns.reduce((acc,fn) => fn(acc), valor)
```

Atente para o operador `...` . Ele se chama **SPREAD** e serve para permitir expandir uma expressão em um local que receba múltiplos argumentos ou elementos. Em outras palavras, ele evita a necessidade de sabermos de antemão a quantidade exata de argumentos que serão passados à função.

[EXEMPLO] Crie um programa para contar o número de caracteres de cada palavra de uma lista, multiplicar esse valor por 3 e depois gerar uma lista crescente dessa lista resultante. Você deve compor as funções de *contagem*, de *triplicação* e de *ordenação* em uma única função.

```
const contagem = (lista) => lista.map((texto) => texto.length)
const triplo = (lista) => lista.map((x) => 3*x)
const ordena = (lista) => lista.sort((a,b) => a-b)

const composicao = (...fns) => (lista) => fns.reduce((acc,fn) => fn(acc), lista)

const geraResultado = composicao(
    contagem,
    triplo,
    ordena
)

const nomes = ['Ana Beatriz', 'Bia', 'Guilherme', 'João', 'Rafael']

console.log(geraResultado(nomes))
```