PROVA 2 - PROGRAMAÇÃO FUNCIONAL, 2022.1, Prof. Hendrik Macedo

Aluno: GABARITO, Turma: T04, Data: 27/09/2022

1. A função abaixo pretende contar quantos números negativos existem numa lista..

[3pt] Complete os slots corretamente tanto no corpo da função quanto na hora de aplica-la.

RESPOSTA: LINHAS EM DESTAQUE NO CÓDIGO.

```
    const conta = (elem) => (lista) => {
        const temp1 = lista.map((x)=>x.toString())
3       const temp2 = temp1.reduce(       (acc,x)=>acc+x,''      )
        const temp3 = temp2.split('')
        const temp4 = temp3.filter((x)=>x==elem)
        return temp4.length
    }
8   console.log(conta(     '-'    )([3,-9.5,77,0,-2,19.3])) //imprime 2
9   console.log(conta(     '-'    )([3,0,1])) //imprime 0
```

EXPLICAÇÃO:

1. O map gera uma nova lista com os items em formato de string: ['3','-9.5','77','0','-2','19.3']
2. O reduce concatenaria todas as strings gerando uma string única: '3-9.5770-219.3'
3. O split reorganiza em lista novamente só que agora individualizando cada caractere: ['3','-','9','.','5','7','7','0','-','2','1','9','.','3']
4. O filter gera uma lista reduzida contendo apenas os algarismos iguais ao passado como argumento.... e se o argumento a ser passado for o caractere '-' (sinal de menos), teríamos: ['-','-']
5. Por fim, basta retornar o tamanho dessa lista reduzida, ou seja, o número de sinal de menos encontrados.
   A grande sacada aqui é usar a transformação para string para contar o número de sinal d emenos que existem na lista!

2. Observe o programa abaixo.

[2pt] O que será impresso com sua execução?

[2pt] Que princípio do Paradigma Funcional está envolvido nesta questão?

```
const processa = (lista) => {
    const c = lista
    c[0]=c[1]
    c[1]=c[2]
    c[2]=c[3]
    c[3]=c[0]
    return c
}
const exemplo = [-3, 4, 1, 8]
console.log(processa(exemplo))
console.log(exemplo)
```

RESPOSTA 1: Será impresso exatamente

```
[4,1,8,4]
[4,1,8,4]
```

RESPOSTA 2: Princípio da IMUTABILIDADE.

EXPLICAÇÃO:

A primeira impressao é o resultado da aplicação da função `processa` à lista [-3,4,1,8]. O que a função faz é substituir o valor da primeira posição da lista pelo valor da segunda (Passo 1), valor da segunda pelo da terceira (Passo 2), valor da terceira pelo da quarta (Passo 3) e o valor da quarta pelo da primeira (Passo 4). Mas observe que isso é feito exatamente nessa ordem descrita acima.
Assim, temos:

```
Inicialmente: c = [-3,4,1,8]
Passo 1: c[0]=c[1] ===> c = [4,4,1,8]
Passo 2: c[1]=c[2] ===> c = [4,1,1,8]
Passo 3: c[2]=c[3] ===> c = [4,1,8,8]
Passo 4: c[3]=c[0] ===> c = [4,1,8,4]
```

Portanto, o resultado de `processa` é `[4,1,8,4]`.

Lembre que a constante `c` não faz uma cópia da lista passada (`exemplo`); ela modifica a lista `exemplo` pois listas em Javascript NÃO SÃO IMUTÁVEIS por natureza. Portanto, a segunda impressão é absolutamente igual à primeira.

3. A função `aprovados` a seguir é responsável por calcular o número de alunos aprovados em uma turma. A turma é uma lista de alunos e cada aluno é uma lista de notas, como pode ser visto no exemplo `turma`.
[2pt] Você deve representar a função `aprovados` como sendo uma aplicação da função `exec`, decidindo sabiamente quais seus argumentos e em que ordem devem ser passados (você só pode usar as funções definidas abaixo e não pode alterar nenhuma delas).
[1pt] Que princípio do Paradigma Funcional salta aos olhos neste problema?

RESPOSTA 1: LINHA EM DESTAQUE NO CÓDIGO.

```
const exec = (...params) => (l) => params.reduce((acc,fn) => fn(acc), l)
const m1 = (lista) => lista.map((x) => x<5?0:1)
const r = (lista) => lista.reduce((acc,x) => acc+x,0)
const f = (lista) => lista.filter((x) => x>0)
const m2 = (lista) => lista.map((x) => x/lista.length)
const m3 = (lista) => lista.map((x)=>m2(x))
const m4 = (lista) => lista.map((x)=>r(x))

const aprovados = exec(m3,m4,m1,r) //ou (m3,m4,m1,f,r) ou (m3,m4,f,m1,r) também funcionam!

const turma = [[5,7,8,0], [10,0,0], [7,8], [2,3,5]]

console.log(aprovados(turma)) //resultado será `2`
```

EXPLICAÇÃO:

1. A função `m3` mapeia cada item da lista externa (ou seja, cada lista interna, notas de um aluno) usando a função `m2`. A função `m2`, por sua vez, mapeia cada item dessa lista mais interna para o mesmo valor dividido pelo número de itens da lista. Fazendo isso, conseguimos facilitar o cálculo da média. Exemplo da aplicação de `m3`:
$[[5,7,8,0],[10,0,0]] \implies [[\frac{5}{4},\frac{7}{4},\frac{8}{4},\frac{0}{4}],[\frac{10}{3},\frac{0}{3},\frac{0}{3}]]$.
2. A seguir, a função `m4` mapeia cada item da lista externa (ou seja, cada lista interna, notas de um aluno divididas pela quantidade) usando a função `r`. A função `r`, por sua vez, faz a redução dos valores de cada lista interna efetuando a soma: $[[\frac{5}{4},\frac{7}{4},\frac{8}{4},\frac{0}{4}],[\frac{10}{3},\frac{0}{3},\frac{0}{3}]] \implies [\frac{20}{4},\frac{10}{3}]$. Eis que temos a média para cada aluno no fim das contas.

3. A função `m1`, agora, mapeia cada um desses valores resultantes para ZERO ou para UM a depender se é um valor abaixo de 5 ou não: $[\frac{20}{4}, \frac{10}{3}] \implies [1, 0]$. Isso faz com que associemos a UM todos os alunos com média acima do mínimo da instituição e com ZERO, os abaixo.
4. Finalmente, a função `r` reduz essa lista usando a soma: $[1, 0] \implies 1$. Ou seja, apenas um aluno foi aprovado no exemplo usado nessa explicação.

RESPOSTA 2: O princípio que salta aos olhos pode ser FUNÇÃO COMO CIDADÃ DE PRIMEIRA CLASSE (já que o código ilustra o uso de função da mesma forma que se usa uma constante qualquer) ou FUNÇÕES DE ALTA ORDEM (já que funções são passadas como argumento de outra função).