Projeto Marvin

FASE 05

Uma lista de exercícios

Capítulo I

Instruções

- Somente esse material serve como referência. Não acredite em rumores.
- Fique atento, esse material pode mudar a qualquer hora antes da submissão.
- Os exercícios são cuidadosamente organizados em ordem de dificuldade, do mais fácil para o mais difícil. Esse mesmo raciocínio é aplicado para as correções, portanto não adianta completar om exercício mais difícil se os anteriores estão errados.
- Tenha certeza que possui as permições necessárias para acessar arquivos e comandos.
- Você precisa seguir os procedimentos de envio para todas as atividades.
- Suas atividades serão corrigidas e avaliadas pelos seus colegas.
- Exercícios em Shell devem ser funcionais através do /bin/bash
- Você não pode deixar nenhum outro arquivo na pasta, a não ser o que foi expressamente pedido.
- Tem uma pergunta? Pergunte ao seu colega à direita.Caso contrário, tente o seu colega à esquerda.
- Seu guia de referência se chama Google / man / Internet
- Procure por conteúdo no Youtube para introduzir ao assunto.
- Examine os exemplos cuidadosamente. Eles estão sempre corretos e podem pedir por detalhes que não foram mencionados explicitamente na atividade.
- A função não deve ser chamada no arquivo, apenas definida.
- Os arquivos devem ter compatibilidade
- “No início, o universo foi criado. Isso irritou profundamente muitas pessoas e, no geral, foi encarado como uma péssima idéia”.

Capítulo II

Exercício 0: Aprendendo a contar

_M	Exercício: 00
Aprendendo a contar	
Diretório de submissão: fase05/ex00	
Arquivo a ser submetido: umADez.js	

- Escreva uma função chamada “umADez”;
- A função deve apenas retornar uma array com os números de 1 a 10;

Capítulo III

Exercício 01: Precisa medir isso

_M	Exercício: 01
Precisa medir isso	
Diretório de submissão: fase05/ex01	
Arquivo a ser submetido: tamanho.js	

- Escreva uma função chamada “tamanho”, que sempre recebe uma array como argumento;
- A função deve retornar o número de itens que a array contém;

```
function tamanho(array) {
...
}
conosole.log(tamanho([“a”, true, 12]))
```

ao ser executado:

```
$>3
```

Capítulo IV

Exercício 02: Quem é o primeiro?

_M	Exercício: 02
Quem é o primeiro?	
Diretório de submissão: fase05/ex02/	
Arquivo a ser submetido: primeiro.js	

- Escreva uma função chamada “primeiro”, que sempre recebe uma array com um ou mais itens como argumento;
- A função deve retornar o primeiro item da array;

Capítulo V

Exercício 03: Lá no final

_M	Exercício: 03
Lá no final	
Diretório de submissão: fase05/ex03/	
Arquivo a ser submetido: penultimo.js	

- Crie um função chamada “penultimo”, que sempre recebe uma array com dois ou mais itens;
- A função deve retornar o penúltimo elemento da array;

Capítulo VI

Exercício 04: Segundo lugar

_M	Exercício: 04
Segundo lugar	
Diretório de submissão: fase05/ex04/	
Arquivo a ser submetido: segundo.js	

- Escreva uma função chamada “segundo”. Essa função sempre vai receber uma array com dois ou mais itens como argumento;
- A função deve retornar a array recebida, adicionando “Marvin” na segunda posição;

```
function segundo(array) {
...
}
conosole.log(segundo([42, “xyz”, true, 0, 14]))
```

ao ser executado:

```
$> [42, “Marvin”, “xyz”, true, 0, 14]
```

Capítulo VII

Exercício 05: Tire isso daqui

_M	Exercício: 05
Tire isso daqui	
Diretório de submissão: fase05/ex05/	
Arquivo a ser submetido: removerPosicao.js	

- Escreva uma função chamada “removerPosicao”, que sempre recebe uma array e um número como argumento;
- O número recebido como argumento sempre será maior que zero e menor que o tamanho da array;
- A função deve retornar a função recebida, removendo o item na posição passada como argumento, a partir do 1;

```
function removerPosicao(array, posicao) {
...
}
conosole.log(removerPosicao([“jantar”, “sala”, mesa”], 1))
```

ao ser executado:

```
$> [“sala”, “mesa”]
```

```
function removerPosicao(array, posicao) {
...
}
conosole.log(removerPosicao([43, 902, 4, -1, 34, 9, 2], 4))
```

ao ser executado:

```
$> [43, 902, 4, 34, 9, 2]
```

Capítulo VIII

Exercício 06: Classificação

_M	Exercício: 06
Classificação	
Diretório de submissão: fase05/ex06/	
Arquivo a ser submetido: classificar.js	

- Escreva uma função chamada “classificar”, que sempre recebe uma array com dois ou mais itens do tipo string como argumento;
- A função deve retornar uma array com os itens da original classificados em ordem alfabética;
- Deve seguir a ordem Unicode, que é a padrão do JavaScript;

```
function classificar(array) {
...
}
conosole.log(classificar([“astro”, “Ferro”, “0”]))
```

ao ser executado:

```
$> [“0”, “astro”, “Ferro”]
```

Capítulo IX

Exercício 07: Quantas letras tem aqui

_M	Exercício: 07
Quantas letras tem aqui	
Diretório de submissão: fase05/ex07/	
Arquivo a ser submetido: quantasLetras.js	

- Escreva uma função chamada “quantasLetras”, que sempre recebe uma array com itens do tipo string;
- A função deve retornar a soma quantidade de caracteres de todos os itens;

```
function quantasLetras(array) {
...
}
conosole.log(quantasLetras([“zap”, “7 de copas”, “dois”]))
```

ao ser executado:

```
$>17
```

Capítulo X

Exercício 08: Varrendo o lixo

_M	Exercício: 08
Varrendo o lixo	
Diretório de submissão: fase05/ex08/	
Arquivo a ser submetido: apenasPalavras.js	

- Escreva uma função chamada “apenasPalaras”, que sempre recebe uma array;
- A função deve retornar uma array com apenas os itens do tipo string da array origina, na mesma ordem;

```
function apenasPalavras(array) {
...
}
conosole.log(apenasPalavras([42, “xyz”, true, 0, “um”]))
```

ao ser executado:

```
$> [“xyz”, “um”]
```