Introdução

Em JavaScript, um array é uma estrutura parecida com uma lista, sem tamanho fixo, que pode conter qualquer tipo de valor primitivo ou objetos, inclusive tipos misturados.

Para criar um array, adicione os elementos entre colchetes []. Para acessar um valor do array, use o índice entre colchetes [] após o identificador. Os índices de um array começam em zero.

Por exemplo:

```
const numbers = [1, 'two', 3, 'four'];
numbers[2];
// => 3
```

Para obter o número de elementos em um array, use a propriedade length:

```
const numbers = [1, 'two', 3, 'four'];
numbers.length;
// => 4
```

Para alterar um elemento do array, atribua um valor ao índice desejado:

```
const numbers = [1, 'two', 3, 'four'];
numbers[0] = 'one';
numbers;
// => ['one', 'two', 3, 'four']
```

Métodos

Alguns métodos disponíveis em todo objeto Array podem ser usados para adicionar ou remover elementos. Aqui estão alguns importantes para este exercício:

push

Um valor pode ser adicionado ao final do array usando .push(valor). O método retorna o novo tamanho do array.

```
const numbers = [1, 'two', 3, 'four'];
numbers.push(5); // => 5
numbers;
// => [1, 'two', 3, 'four', 5]
```

pop

Remove o último valor do array usando .pop(). O método retorna o valor removido. O tamanho do array diminui.

```
const numbers = [1, 'two', 3, 'four'];
numbers.pop(); // => 'four'
numbers;
// => [1, 'two', 3]
```

shift

Remove o primeiro valor do array usando .shift(). O método retorna o valor removido.

```
const numbers = [1, 'two', 3, 'four'];
numbers.shift(); // => 1
numbers;
// => ['two', 3, 'four']
```

unshift

Adiciona um valor no início do array usando .unshift(valor). O método retorna o novo tamanho do array.

```
const numbers = [1, 'two', 3, 'four'];
numbers.unshift('one'); // => 5
numbers;
// => ['one', 1, 'two', 3, 'four']
```

splice

Remove um valor em um índice específico usando .splice(índice, 1). O método retorna os elementos removidos.

```
const numbers = [1, 'two', 3, 'four'];
numbers.splice(2, 1, 'one'); // => [3]
numbers;
// => [1, 'two', 'one', 'four']
```

Avançado

Esses métodos são mais poderosos do que o descrito acima:

- push e unshift permitem adicionar vários valores de uma vez.
- splice pode remover vários valores aumentando o segundo argumento.
- splice também pode adicionar vários valores após o deleteCount, permitindo substituir ou inserir elementos no meio do array.

Esses recursos **não são necessários** para completar este exercício.

Instruções

Como uma aspirante a maga, Elyse precisa praticar o básico. Ela tem uma pilha de cartas que quer manipular.

Para facilitar, ela usa apenas cartas de 1 a 10, então sua pilha pode ser representada por um array de números. A posição de cada carta corresponde ao índice do array. Isso significa que a posição 0 se refere à primeira carta, a posição 1 à segunda, e assim por diante.

Observação

Todas as funções, **exceto duas**, devem atualizar o array de cartas e retornar o array modificado — um padrão comum conhecido como **Builder pattern**, que permite encadear funções facilmente.

As duas exceções são:

- `getItem`: deve retornar a carta na posição informada
- `checkSizeOfStack`: deve retornar `true` se o tamanho informado for igual ao da pilha

---

Exercícios

1. Recuperar uma carta da pilha

Para pegar uma carta, retorne a carta no índice position da pilha fornecida.

```
const position = 2;
getItem([1, 2, 4, 1], position);
// => 4
```

---

2. Trocar uma carta da pilha

Faça um truque e troque a carta no índice position pela carta de substituição fornecida. Retorne a pilha ajustada.

```
const position = 2;
const replacementCard = 6;
setItem([1, 2, 4, 1], position, replacementCard);
// => [1, 2, 6, 1]
```

---

3. Inserir uma carta no topo da pilha

Faça uma carta aparecer inserindo uma nova carta no topo da pilha. Retorne a pilha ajustada.

```
const newCard = 8;
insertItemAtTop([5, 9, 7, 1], newCard);
// => [5, 9, 7, 1, 8]
```

---

4. Remover uma carta da pilha

Faça uma carta desaparecer removendo a carta na posição informada. Retorne a pilha ajustada.

```
const position = 2;
removeItem([3, 2, 6, 4, 8], position);
// => [3, 2, 4, 8]
```

---

5. Remover a carta do topo da pilha

Faça uma carta desaparecer removendo a carta do topo da pilha. Retorne a pilha ajustada.

```
removeItemFromTop([3, 2, 6, 4, 8]);
// => [3, 2, 6, 4]
```

---

6. Inserir uma carta no fundo da pilha

Faça uma carta aparecer inserindo uma nova carta no fundo da pilha. Retorne a pilha ajustada.

```
const newCard = 8;
insertItemAtBottom([5, 9, 7, 1], newCard);
// => [8, 5, 9, 7, 1]
```

---

7. Remover uma carta do fundo da pilha

Faça uma carta desaparecer removendo a carta do fundo da pilha. Retorne a pilha ajustada.

```
removeItemAtBottom([8, 5, 9, 7, 1]);
// => [5, 9, 7, 1]
```

---

8. Verificar o tamanho da pilha

Verifique se o tamanho da pilha é igual a stackSize.

```
const stackSize = 4;
checkSizeOfStack([3, 2, 6, 4, 8], stackSize);
// => false
```