Introdução

O **for loop** é uma das estruturas mais usadas para executar repetidamente uma lógica. Em JavaScript, ele consiste na palavra-chave `for`, um cabeçalho entre parênteses e um bloco de código que contém o corpo do loop, envolvido por chaves.

```
for (initialization; condition; step) {
  // código que é executado repetidamente enquanto a condição for verdadeira
}
```

A **inicialização** geralmente define uma variável contadora, a **condição** verifica se o loop deve continuar ou parar, e o **passo** incrementa o contador ao final de cada repetição. As partes do cabeçalho são separadas por ponto e vírgula.

```
const list = ['a', 'b', 'c'];
for (let i = 0; i < list.length; i++) {
  // código que deve ser executado para cada item list[i]
}
```

Definir o passo geralmente é feito usando os operadores de **incremento** ou **decremento** do JavaScript, como mostrado no exemplo acima. Esses operadores modificam a variável diretamente.
`++` adiciona 1 a um número e `--` subtrai 1 de um número.

```
let i = 3;
i++;
// i agora é 4

let j = 0;
j--;
// j agora é -1
```

Instruções

Você é um observador de pássaros apaixonado e acompanha quantos pássaros visitaram seu jardim. Normalmente, você usa anotações em um caderno para contar os pássaros, mas agora quer trabalhar melhor com esses dados. Você já digitalizou a contagem diária de pássaros das últimas semanas que estavam anotadas no caderno.

Agora você quer determinar o número total de pássaros observados, calcular a contagem de uma semana específica e corrigir um erro de contagem.

Nota

Para praticar, use um **for loop** para resolver cada uma das tarefas abaixo.

---

1. Determinar o número total de pássaros observados até agora

Vamos começar analisando os dados de forma geral. Descubra quantos pássaros você contou no total desde que começou os registros.

Implemente uma função `totalBirdCount` que receba um objeto semelhante a um array contendo a contagem de pássaros por dia. Ela deve retornar o número total de pássaros observados.

```
birdsPerDay = [2, 5, 0, 7, 4, 1, 3, 0, 2, 5, 0, 1, 3, 1];
totalBirdCount(birdsPerDay);
// => 34
```

---

2. Calcular o número de pássaros visitantes em uma semana específica

Agora que você já tem uma noção geral dos números, quer fazer uma análise mais detalhada.

Implemente uma função `birdsInWeek` que receba um objeto semelhante a um array com a contagem diária de pássaros e um número da semana. Ela deve retornar o número total de pássaros observados naquela semana específica. Você pode assumir que as semanas sempre são registradas completamente.

```
birdsPerDay = [2, 5, 0, 7, 4, 1, 3, 0, 2, 5, 0, 1, 3, 1];
birdsInWeek(birdsPerDay, 2);
// => 12
```

---

3. Corrigir um erro de contagem

Você percebeu que, durante todo o tempo em que estava observando os pássaros, havia um deles escondido em um canto distante do jardim. Você descobriu que esse pássaro sempre aparecia **a cada dois dias** no seu jardim. Você não sabe exatamente onde ele estava nos dias intermediários, mas com certeza não estava no seu jardim. Sua intuição de observador também diz que esse pássaro estava presente **no primeiro dia** registrado na lista.

Com essa nova informação, escreva uma função `fixBirdCountLog` que receba um objeto semelhante a um array com a contagem diária de pássaros. Ela deve corrigir o erro de contagem **modificando o array original**.

```
birdsPerDay = [2, 5, 0, 7, 4, 1];
fixBirdCountLog(birdsPerDay);
// => [3, 5, 1, 7, 5, 1]
```