Checkpont parte 2

Abaixo estão 5 exercícios, escolher **somente 2** e desenvolver a logica com todos os cenários de testes.
Serão analisados solução da resposta e assertividade dos cenários de testes. Quando mais cenários melhor. Pelo menos dois testes unitários, cenário feliz e um cenário de excessão.

Pode fazer mais exercícios isso contribui para a nota final ser mais alta.
Para essa atividade não é necessário uso de mock

Atividade individual.

Somente teste unitário. Pode ser maven ou gradle. Spring boot

Enviar link do github até 30/09/23 11:59 AM.

Nota máxima 5

1. Verificador de Parênteses

Descrição: Escreva uma função que recebe uma string contendo apenas os caracteres '(', ')', '{', '}', '[', e ']' e determine se a entrada é válida. A entrada é válida se:

Os parênteses abertos são fechados pela mesma ordem de parênteses que os abriu.
Os parênteses abertos são fechados na ordem correta.
Endpoint: /validate-parentheses
Método: POST
Entrada: Um objeto JSON com uma chave "input" e o valor sendo a string para validar.
Retorno: Um objeto JSON com a chave "isValid" e o valor sendo true ou false.

```
@PostMapping("/validate-parentheses")
public boolean validateParentheses(@RequestBody
String input) {
  //resposta
}
```

Entrada: {[()]}
Retorno: true

Entrada: {[(])}
Retorno: false

2. Caminho mais curto em um labirinto

Descrição: Dado um labirinto representado como uma matriz de 0s e 1s, encontre o caminho mais curto de um ponto de partida a um ponto de destino. Aqui, 0 representa um caminho aberto e 1 representa um muro. Você pode se mover para cima, para baixo, para a esquerda ou para a direita. Retorne o menor número de passos necessários para ir do ponto de partida ao ponto de destino.

Endpoint: /shortest-path
Método: POST
Entrada: Um objeto JSON contendo as chaves "maze" (uma matriz de 0s e 1s) e "start" (um ponto de partida) e "end" (um ponto de destino).
Retorno: Um objeto JSON com a chave "steps" e o valor sendo o número de passos.

```
@PostMapping("/shortest-path")
public int shortestPath(@RequestBody MazeRequest
request) {


  public class MazeRequest {
    private int[][] maze;
    private Point start;
    private Point end;
    // getters and setters...
  }

  public class Point {
    private int x, y;
    // constructors, getters and setters...
  }
```

```
{
  {0, 1, 0, 0},
  {0, 0, 0, 0},
  {1, 1, 1, 0},
  {1, 1, 0, 0}
}
Start: (0,0)
End: (3,3)
```

Caminho Possível: Comece em (0,0), mova para a direita até (0,3), depois desça até (3,3).

Retorno: 6 (contando os passos: direita, direita, direita, para baixo, para baixo, para baixo).

Neste exemplo, o caminho do início (0,0) ao final (3,3) é claro e não é bloqueado por muros. Portanto, a resposta é 6 passos.

3. Sub array com soma máxima

Descrição: Dado um array de números inteiros (que pode conter números negativos), encontre o subarray contíguo que tem a maior soma e retorne essa soma.

Endpoint: /max-subarray
Método: POST
Entrada: Um objeto JSON com uma chave "array" e o valor sendo a lista de números inteiros.
Retorno: Um objeto JSON com a chave "maxSum" e o valor sendo a soma máxima.

```
@PostMapping("/max-subarray")
public int maxSubArraySum(@RequestBody int[]
nums) {
}
```

Entrada: [-2, 1, -3, 4, -1, 2, 1, -5, 4]
Retorno: 6 (porque o subarray [4, -1, 2, 1] tem a maior soma)

4. Encontre o número que aparece uma única vez

Descrição: Dado um array não vazio de números inteiros, todo elemento aparece duas vezes exceto por um. Encontre esse único número.

```
@PostMapping("/single-number")
public int findSingleNumber(@RequestBody int[]
nums) {
}
```

Entrada: [4, 1, 2, 1, 2]
Retorno: 4

5. Número Máximo de Consecutivos

Descrição: Dado um array binário, encontre a máxima sequência de uns consecutivos.

```
@PostMapping("/max-consecutive-ones")
public int findMaxConsecutiveOnes(@RequestBody
int[] nums) {
}
```

Entrada: [1, 1, 0, 1, 1, 1]
Retorno: 3

7. Verificar Palíndromo

Descrição: Crie uma função que verifique se uma determinada string é um palíndromo. Um palíndromo é uma palavra, frase, número ou outra sequência de caracteres que lê o mesmo para a frente e para trás (ignorando espaços, pontuação e capitalização).

```
@PostMapping("/is-palindrome")
public boolean isPalindrome(@RequestBody String
s) {

}
```

Exemplo:
Entrada: "A man, a plan, a canal, Panama"
Saída: Verdadeiro

Entrada: "hello"
Saída: Falso