Teste Símios - Mercado Livre

Em um futuro distante, na cadeia de evolução, os símios e os humanos estão cada vez mais próximos. Por esse motivo ficou muito difícil distinguir quem é humano e quem é símio.

Você é um cientista contratado para desenvolver um projeto que detecta se uma sequência de DNA pertence a um humano ou a um símio.

Para isso, você precisar desenvolver um programa, com um método ou função com a seguinte assinatura (em uma das seguintes linguagens: **(Java** / **Golang** / **Javascript (Node)** / **Python)**

```
boolean isSimian (String[] dna)  // Exemplo em Java
```

Você receberá como **parâmetro** um **array de Strings** que representam **cada linha de uma tabela quadrada de (NxN)** com a sequência de DNA.

A	T	G	C	G	A
C	A	G	T	G	C
T	T	A	T	T	T
A	G	A	C	G	G
G	C	G	T	C	A
T	C	A	C	T	G

Humano

C	T	G	A	G	A
C	T	A	T	G	C
T	A	T	T	G	T
A	G	A	G	G	G
C	C	C	C	T	A
T	C	A	C	T	G

Símio

As letras da String **só podem ser: (A, T, C, G)**, que representa cada base nitrogenada do DNA.

Você saberá se um DNA pertence a um símio, se encontrar **mais de uma sequência de quatro letras iguais** nas direções horizontais, verticais ou nas diagonais.

Exemplo (Símio):
String [] dna = {"CTG**AG**A", "CTG**AG**C", "T**AT**T**G**T", "**A**GAG**G**G", "**CCCC**TA", "TCACTG"};

Nesse caso, a chamada para a função **`isSimian(String[] dna)`** deve retornar "true".

Com bases nessas informações, desenvolva o algoritmo da maneira mais eficiente possível de acordo com os desafios abaixo

Desafios

Nível 1:

Desenvolva um método ou função que esteja de acordo com a assinatura proposta **`isSimian(String[] dna)`**, que seja capaz de identificar corretamente símios.

Nível 2:

Crie uma **API REST** e hospede em algum ambiente de computação em **nuvem gratuita** (Google App Engine, Amazon AWS, etc).
Você deve disponibilizar **um endpoint "/simian".** Esse serviço recebe uma sequência de DNA através de um **HTTP POST** com um **JSON** que contém o seguinte formato, exemplo:

```
POST → /simian
{
      "dna": ["ATGCGA", "CAGTGC", "TTATGT", "AGAAGG", "CCCCTA", "TCACTG"]
}
```

Caso o DNA seja identificado como um **símio**, você deve retornar um **HTTP 200-OK**, caso contrário um **HTTP 403-FORBIDDEN**

Nível 3:

Crie um banco de dados, que **armazena** os DNAs verificados pela API. Esse banco deve garantir a unicidade, ou seja, **apenas 1 registro por DNA.**

Disponibilizar um serviço extra **"/stats"** que responde um **HTTP GET.** A resposta deve ser um Json que retorna as estatísticas de verificações de DNA, onde deve informar a **quantidade de DNA's símios, quantidade de DNA's humanos, e a proporção de símios para a população humana.**
Segue exemplo da resposta:

```
{"count_mutant_dna": 40, "count_human_dna": 100: "ratio": 0.4}
```

O que entregar

- Código-fonte (**Para os níveis 2 e 3**).
    - Criar um repositório **privado** no Github ou Bitbucket
    - Após finalizar o teste adicionar o usuário **ITMLB** como colaborador para que possamos ter acesso ao código.
    - Se o repositório estiver público, será automaticamente desqualificado.
- Instruções sobre como executar o programa ou a API. (**Para os níveis 2 e 3** em README).
- URL da API (**Para os níveis 2 e 3**)

Observações:

- Tenha em mente que faremos uma série de testes (**Para os níveis 2 e 3 POSTs e GETs)** com matrizes válidas e inválidas.
- Considere a performance do algoritmo e o tempo de resposta da aplicação (**Para os níveis 2 e 3)**, sabendo que a API **pode receber flutuações** de tráfego **agressivas**
- O projeto deve conter **testes automáticos**, e a **cobertura do código** deve ser **> 80%**.