Combinações de resultados

Prova Backend

Instruções

- ➔ Prazo de sete dias corridos.
- ➔ Entregar projeto utilizando docker e se possível scripts de automatização para rodar a aplicação e testes.
- ➔ Linguagens: Go, PHP, Python, Java, Ruby, Node.JS, C#.
- ➔ Enviar projeto zipado por email.
- ➔ A solução para o problema proposto deve ser feito pelo candidato.
- ➔ Pode utilizar bibliotecas para testes e API em REST ou GraphQL.

Critérios de avaliação

- ➔ Solução (adequação aos requisitos do problema)
- ➔ Complexidade Big O
- ➔ Testes
- ➔ Organização de projeto
- ➔ Legibilidade do código
- ➔ Simplicidade

O Problema

Joaquim é um jovem estudante que pretende ajudar seus colegas de classe a resolver um problema de conta. Ele pretende construir um API web que resolva o problema passado pela professora e que retorne o resultado para os seus colegas utilizarem. A sua tarefa será ajudar Joaquim nessa missão.

Dado um resultado de placar de jogo de futebol americano, Joaquim precisa calcular o **maior número de combinações** de pontuações possíveis para o resultado daquela partida. A ordem das pontuações não importa, apenas os tipos de pontuações diferentes de cada time. Os pontos podem ser marcados da seguinte forma:

- Touchdown: 6 pontos
- Extra touchdown: 0, 1 ou 2 pontos (só pode ser marcado após um touchdown)
- Field goal: 3 pontos

Por exemplo, para o placar de 3 x 15, há quatro combinações possíveis. O time com 3 pontos só pode ter marcado através de um field goal, enquanto o time com 15 pontos pode ter feito 4 tipos diferentes de jogadas: um touchdown seguido de outro touchdown e um field goal, um touchdown seguido de três field goals, cinco field goals, ou um touchdown seguido de um extra touchdown (1 ponto) e outro touchdown seguido de um extra touchdown (2 pontos).

Placar: 3 x 15
Combinações possíveis: 4

Um exemplo de resultado não possível seria 8 x 5, nesse caso é impossível um time marcar 5 pontos.

Placar: 8 x 5
Combinações possíveis: 0

Entrada / Saída

Para facilitar a consulta dos amigos de Joaquim, você deve receber a string por meio de uma API Web. Ou seja, cada consulta é feita através de uma requisição HTTP que deve retornar o resultado em formato JSON. A sua API pode ser feita seguindo os padrões REST ou GraphQL. Você pode escolher o formato que se sentir mais confortável. Considere que a entrada é sempre válida e deve ser no formato {inteiro}x{inteiro}.

A professora do Joaquim costuma dar pontos extras para APIs em formato GraphQL :)

API REST

A API deve possuir uma única rota **/verify** que recebe uma requisição REST em formato JSON contendo o placar da partida no corpo da requisição. Por exemplo, para a entrada apresentada anteriormente, a requisição seria:

URL: http://localhost:8080/verify
Method: POST

```
{ "score": "3x15" }
```

A resposta deve ser feita também em formato JSON, retornando o número de combinações de pontuações encontrado. No exemplo anteriormente apresentado, o resultado seria:

Content-Type: application/json

```
{
   "combinations": 4
}
```

API GraphQL

A API GraphQL deve possuir uma única rota **/graphql** que recebe uma requisição contendo uma única *mutation* chamada **verify**. Por exemplo, para a entrada apresentada anteriormente, a requisição seria:

URL: http://localhost:8080/graphql
Method: POST

```
mutation {
    verify(score: "3x15") {
        combinations
    }
}
```

A saída da API GraphQL deve respeitar o formato apresentado acima, retornando o resultado por meio do *resolver* apresentado anteriormente, na qual retorna o maior número de combinações encontradas.

Observações

- ➔ Deixe claro como seu programa deve ser executado. Scripts de automatização, testes e Dockerfiles são sempre bem-vindos.
- ➔ Documente sua lógica de implementação para entendermos o máximo possível do seu programa.
- ➔ Considere que as entradas serão sempre válidas.

Boa prova =)