Coding: Caixa Eletrônico

O programa (padrão de stdin e stdout) deve simular algumas funcionalidades referentes a um caixa eletrônico, sendo elas:

1 - Abastecimento do caixa eletrônico (adicionar notas no caixa)
2 - Saque

Deve assumir que:

- Todos os valores numéricos são inteiros e positivos;
- Os saques chegarão no caixa eletrônico em ordem cronológica;
- Os saques serão feitos sempre pela mesma pessoa;
- As entradas em JSON sempre serão válidas, ou seja, não terá necessidade de validar se o json é válido.
- Todas as operações serão sequenciais.

---

1. Abastecimento do caixa eletrônico (adicionar cédulas no caixa)

Entrada

A entrada para abastecimento é

```
# Objeto caixa
{
   "caixa":{
      "caixaDisponivel":boolean,
      "notas":{
         "notasDez":number,
         "notasVinte":number,
         "notasCinquenta":number,
         "notasCem":number
      }
   }
}
```

Saída

O estado atual do caixa junto com quaisquer violações da lógica de negócios. Se não houverem violações no processamento da operação, o campo `erros` deve retornar um vetor vazio `[ ]`.

```
#Saída
{
   "caixa":{
      "caixaDisponivel":boolean,
      "notas":{
         "notasDez":number,
         "notasVinte":number,
         "notasCinquenta":number,
         "notasCem":number
      }
   },
   "erros":[]
}
```

Violações da lógica de negócios

Uma vez disponibilizado para uso o caixa não poderá ter sua quantidade de notas modificada. Se o aplicativo receber uma segunda operação de abastecimento do caixa eletrônico, ele deverá retornar a seguinte violação: **caixa-em-uso**

Cenários de uso:
Cenário 1 : Abastecimento de notas porém com caixa ainda indisponível para realização de operações:

```
#Entrada
{
   "caixa":{
      "caixaDisponivel":false,
      "notas":{
         "notasDez":100,
         "notasVinte":50,
         "notasCinquenta":10,
         "notasCem":30
      }
   }
}
```

```
#Saída
{
   "caixa":{
      "caixaDisponivel":false,
      "notas":{
         "notasDez":100,
         "notasVinte":50,
         "notasCinquenta":10,
         "notasCem":30
      },
   },
   "erros":[]
}
```

Cenário 2: Realizar o abastecimento com a sua disponibilização para uso

```
#Entradas
# Entrada 1: Abastecimento
{
   "caixa":{
      "caixaDisponivel":false,
      "notas":{
         "notasDez":100,
         "notasVinte":50,
         "notasCinquenta":10,
         "notasCem":30
      }
   }
}
# Entrada 2: Abastecimento
{
   "caixa":{
      "caixaDisponivel":true,
      "notas":{
         "notasDez":5,
         "notasVinte":5,
         "notasCinquenta":5,
         "notasCem":15
      }
   }
}
```

```
# Saídas
# Saída 1: Referente ao abastecimento da entrada 1
{
   "caixa":{
      "caixaDisponivel":false,
            "notas":{
               "notasDez":100,
               "notasVinte":50,
               "notasCinquenta":10,
               "notasCem":30
            }
         },
         "erros":[]
}
# Saída 2: Referente ao abastecimento da entrada 2
{
   "caixa":{
      "caixaDisponivel":true,
      "notas":{
          "notasDez":5,
          "notasVinte":5,
          "notasCinquenta":5,
          "notasCem":15
        }
    },
    "erros":[]
}
```

Cenário 3: Abastecimento que viola a lógica do caixa eletrônico

Dado que o caixa já está disponível para uso (caixaDisponivel : true), tentar realizar uma nova operação de abastecimento

```
#Entradas
# Entrada 1: Abastecimento
{
   "caixa":{
      "caixaDisponivel":true,
      "notas":{
         "notasDez":100,
         "notasVinte":50,
         "notasCinquenta":10,
         "notasCem":30
      }
   }
}
# Entrada 2: Abastecimento
{
   "caixa":{
      "caixaDisponivel":true,
      "notas":{
         "notasDez":350,
         "notasVinte":1500,
         "notasCinquenta":2500,
         "notasCem":100
      }
   }
}
```

```
# Saídas
# Saída 1: Referente ao abastecimento 1
{
   "caixa":{
      "caixaDisponivel":true,
      "notas":{
         "notasDez":100,
         "notasVinte":50,
         "notasCinquenta":10,
         "notasCem":30
      }
   },
  "erros":[]
}
# Saída 2: Referente ao abastecimento 2
{
   "caixa":{
      "caixaDisponivel":true,
      "notas":{
         "notasDez":100,
         "notasVinte":50,
         "notasCinquenta":10,
         "notasCem":30
      }
   },
   "erros":[“caixa-em-uso”]
}
```

2. Saque

O saque no caixa eletrônico deve considerar sempre entregar o menor número de notas possíveis.

Exemplo:
Exemplo 1: Considerando que teremos notas suficientes de todos os valores
Valor do saque 30 - Espera-se retirar no caixa eletrônico 1 nota de 20 e 1 nota de 10.
Valor do saque 80 - Espera-se retirar no caixa eletrônico 1 nota de 50, 1 nota de 20 e 1 nota de 10
Exemplo 2: **Considerando que não temos notas de 20**
Valor do saque 30 - Espera-se retirar no caixa eletrônico 3 notas de 10.
Valor do saque 80 - Espera-se retirar no caixa eletrônico 1 nota de 50 e 3 notas de 10

Entrada do saque
Objeto saque contendo o valor e horário do saque.

```
#Objeto de saque
{
   "saque":{
      "valor":number,
      "horario":datetime
   }
}
```

Saida do saque

O estado atual do caixa junto com quaisquer violações da lógica de negócios. Se não houverem violações no processamento da operação, o campo `erros` deve retornar um vetor vazio `[ ]`.

```
#Saída
{
   "caixa":{
      "caixaDisponivel":boolean,
      "notas":{
         "notasDez":number,
         "notasVinte":number,
         "notasCinquenta":number,
         "notasCem":number
      }
   },
   "erros":[]
}
```

Violações da lógica de negócios

- Estrutura do caixa inexistente: erro - **caixa-inexistente**
- Caixa indisponível para realização de operações: **caixa-indisponivel**
- Valor de saque maior do que a quantidade de dinheiro disponível: **valor-indisponivel**
- Não deve haver mais de um saque bem-sucedido com o mesmo valor em um intervalo inferior a 10 minutos: **saque-duplicado**.

Cenários de exemplo:

1 - Realizando um saque com sucesso: Dado um caixa abastecido (com quantidade suficiente de notas para qualquer tipo de saque) e disponível para uso.

```
#Entrada
# Entrada 1 - Abastecimento
{
   "caixa":{
      "caixaDisponivel":true,
      "notas":{
         "notasDez":100,
         "notasVinte":50,
         "notasCinquenta":10,
         "notasCem":30
      }
   }
}
#Entrada 2 - Saque
{
   "saque":{
      "valor":80,
      "horario":"2019-02-13T11:01:01.000Z"
   }
}
```

```
# Saída
# Saída 1 - Referente ao abastecimento
{
   "caixa":{
      "caixaDisponivel":true,
      "notas":{
         "notasDez":100,
         "notasVinte":50,
         "notasCinquenta":10,
         "notasCem":30
      }
   },
   "erros":[]
}
# Saída 2 - Referente ao saque
{
   "caixa":{
      "caixaDisponivel":true,
      "notas":{
         "notasDez":99,
         "notasVinte":49,
         "notasCinquenta":9,
         "notasCem":30
      }
   },
   "erros":[]
}
```

2 - Caixa inexistente: tentativa de realização de saque.

```
#Entrada
# Entrada 1 - Saque
{
   "saque":{
      "valor":80,
      "horario":"2019-02-13T11:01:01.000Z"
   }
}
```

```
# Saída
# Saída 1 - Referente ao saque
{
   "caixa":{},
   "erros":[“caixa-inexistente”]
}
```

3 - Valor indisponível - tentativa de realização de saque em um caixa que está sem quantidade de dinheiro disponível

```
#Entrada
# Entrada 1 - Abastecimento do caixa
{
   "caixa":{
      "caixaDisponivel":true,
      "notas":{
         "notasDez":0,
         "notasVinte":0,
         "notasCinquenta":1,
         "notasCem":3
      }
   }
}
#Entrada 2 - Saque
{
   "saque":{
      "valor":600,
      "horario":"2019-02-13T11:01:01.000Z"
   }
}
```

```
# Saída
# Saída 1 - Referente ao abastecimento
{
   "caixa":{
      "caixaDisponivel":true,
      "notas":{
         "notasDez":0,
         "notasVinte":0,
         "notasCinquenta":1,
         "notasCem":3
      }
   },
   "erros":[]
}
```

```
# Saída 2 - Referente ao saque
{
   "caixa":{
      "caixaDisponivel":true,
      "notas":{
         "notasDez":0,
         "notasVinte":0,
         "notasCinquenta":1,
         "notasCem":3
      }
   },
   "erros":[“valor-indisponivel”]
}
```

Leia com atenção:

- A aplicação não deve depender de nenhum tipo de armazenamento (banco de dados, arquivo de texto, etc.). Todo o processamento deve ser feito em memória, utilizando qualquer estrutura de dados que seja considerada relevante.
- A aplicação deve conter um arquivo README.md que explique seu funcionamento e a forma de execução de forma clara (obrigatório utilização de estrutura de containers com docker).
- O código fonte não deve ser exposto em nenhum repositório de código público ou compartilhado.
- Favor remover toda e qualquer indicação que possa lhe identificar seu código fonte antes de enviar-nos a solução. A solução deve ser enviada em formato de arquivo compactado (zip)
- Se tiver alguma dúvida, por favor, entre em contato o quanto antes.

Good coding 🙂