Exercícios

1) Crie uma função `impar` que retorna `true` se um `Int` é ímpar.

2) Faça uma função `fatorial` que recebe um parâmetro `Int` e retorna o fatorial dele.

3) Faça uma função `minMax` que recebe um vetor de inteiros e retorne uma tupla opcional com dois inteiros, sendo o primeiro o menor elemento e o segundo, o maior.

Exemplo:

```
minMax([])            // nil
minMax([2,0,-1])      // (-1, 2)
```

4) Faça uma função `busca` que recebe um vetor de inteiros e um inteiro a ser buscado (de forma sequencial). Retorne o primeiro índice do vetor ou nil se o elemento não existir.

Exemplo:

```
busca([1, 0, 5, 10, 5], 8) // nil
busca([1, 0, 5, 10, 5], 5) // 2
```

5) Crie uma função `palindromo` que indica se uma `String` é palíndroma (ou seja, ela é lida da mesma maneira de trás pra frente).

`Exemplo:`

```
palindromo("arara") // true
palindromo("sabiá") // false
```

6) Crie uma função `impares` que, dado um vetor de `Int`, retorne outro vetor com os elementos que são ímpares.

`Exemplo:`

```
impares([])                 // []
impares([0, 2, 4])          // []
impares([1, 2, 3, 5])       // [1, 3, 5]
impares([1, 2, 1, 5])       // [1, 1, 5]
```

7) Vamos jogar jokenpô!

7.1) Crie um enum para representar as possíveis jogadas (pedra, papel ou tesoura)

7.2) Crie outro enum para representar o resultado de uma partida (vitória, derrota ou empate)

7.3) Crie uma função chamada `jokenpo` que recebe duas jogadas e retorna o resultado (na ótica do primeiro jogador, ou seja, do primeiro parâmetro).

`Exemplo:`

```
jokenpo(.Tesoura, .Tesoura) // .Empate
jokenpo(.Pedra, .Papel)     // .Derrota
jokenpo(.Pedra, .Tesoura)   // .Vitoria
```

8) Crie uma função chamada `frequencia` que recebe um vetor de `Int` e retorne um dicionário onde cada chave é um elemento do vetor e o valor associado é sua frequência.

Exemplo:

```
frequencia([1, 2, 3, 2, 3, 5, 2, 1, 3, 4, 2, 2, 2])
// [5: 1, 2: 6, 3: 3, 1: 2, 4: 1]
```

9) Crie um protocolo chamado `Shape` que contém os métodos `area()` e `perimeter()`, que retornam `Double`.

Crie agora as structs `Circle`, `Square` e `Rectangle`, implementando o protocolo `Shape`. Declare as propriedades necessárias para o cálculo de área e perímetro para cada uma das structs.