```gleam
import gleam/int
import gleam/string

// Gleam suporta tipos opacos, que permitem exportar o tipo sem exportar seu detalhes internos
pub opaque type Counter {
    Counter(value: Int, step: Int)
}

// Exemplo de tagged union (ou sum type) - o tipo Message possui 3 diferentes construtores
pub type Message {
    Increment
    Decrement
    Reset
}

// A função init recebe como argumento apenas o parametro inteiro step e retorna um Counter
pub fn init(step: Int) -> Counter {
    Counter(value: 0, step: step)
}

// Na função update nós vamos fazer o handle das mensages e casar a mensagem com a sua função específica
pub fn update(counter: Counter, message: Message) -> Counter {
    case message {
        Increment -> Counter(..counter, value: counter.value + counter.step)
        Decrement -> Counter(..counter, value: counter.value - counter.step)
        Reset -> init(counter.step)
    }
}

// E por fim a função render recebe um Counter e retorna uma string com um label e o valor desse counter
pub fn render(counter: Counter) -> String {
    string.append("The current count is: ", int.to_string(counter.value))
}
```

```gleam
// Curry
// Recebe uma função com aridade 2 e retorna o seu equivalente transformado
pub fn curry2(fun: fn(a, b) -> value) {
  fn(a) { fn(b) { fun(a, b) } }
}

// Gleam suporta nativamente curry com até aridade 6, sendo facilmente expandivel se necessário
pub fn curry3(fun: fn(a, b, c) -> value) {
  fn(a) { fn(b) { fn(c) { fun(a, b, c) } } }
}

pub fn curry4(fun: fn(a, b, c, d) -> value) {
  fn(a) { fn(b) { fn(c) { fn(d) { fun(a, b, c, d) } } } }
}

pub fn curry5(fun: fn(a, b, c, d, e) -> value) {
  fn(a) { fn(b) { fn(c) { fn(d) { fn(e) { fun(a, b, c, d, e) } } } } }
}

pub fn curry6(fun: fn(a, b, c, d, e, f) -> value) {
  fn(a) {
    fn(b) { fn(c) { fn(d) { fn(e) { fn(f) { fun(a, b, c, d, e, f) } } } } }
  }
}

// Meu curry com aridade 7
pub fn curry7(fun: fn(a, b, c, d, e, f, g) -> value) {
  fn(a) {
    fn(b) { fn(c) { fn(d) { fn(e) { fn(f) { fn(g) { fun(a, b, c, d, e, f, g) } } } } } }
  }
}
```