```gleam
pub fn curry2_test() {
  let fun = fn(a, b) { a + b }
  let curried = gleam_lab.curry2(fun)

  curried(1)(2)
  |> should.equal(3)
}

pub fn curry3_test() {
  let fun = fn(a, b, c) { a + b - c}
  let curried = gleam_lab.curry3(fun)

  curried(1)(2)(3)
  |> should.equal(0)
}

pub fn curry4_test() {
  let fun = fn(a, b, c, d) { a + b + c + d}
  let curried = gleam_lab.curry4(fun)

  curried(1)(1)(1)(1)
  |> should.equal(4)
}

// Teste para o meu curry de aridade 7
pub fn curry7_test() {
  let fun = fn(a, b, c, d, e, f, g) { a + b + c + d + e + f + g}
  let curried = gleam_lab.curry7(fun)

  curried(1)(1)(1)(1)(1)(1)(1)
  |> should.equal(7)
}
```

```gleam
// Composição de funções
// Recebe duas funções e as encadeia para formar uma função que recebe
// a entrada da primeira e retorna a saída da segunda
pub fn compose(fun1: fn(a) -> b, fun2: fn(b) -> c) -> fn(a) -> c {
  fn(a) { fun2(fun1(a)) }
}

pub fn add_one_and_double() {
  let add_one = fn(x: Int) {x + 1}
  let double = fn(x: Int) {x * 2}

  compose(add_one, double)
}
```

```gleam
pub fn add_one_and_double_test() {
  1
  |> gleam_lab.add_one_and_double()
  |> should.equal(4)
}
```