Aluno: Pedro Mazzurana Cibulski
RA: 21149375-2

Questão 1 - [1 ponto] - Explique a diferença entre funções puras e funções de ordem superior em programação funcional. Dê exemplos de cada uma.
R: Funções puras são funções em que cada uma entrada recebe uma saída como uma simples função de matemática multiplicada por dois. Já as funções de ordem superior, podem receber callbacks, podem retornar outras funções, seria algo mais "livre", um exemplo seria uma função para fazer um Fibonacci com Elixir.

Questão 2 - [1 ponto] - Discorra sobre as vantagens da linguagem de programação elixir e em que tipo de projeto ou cenário devemos optar pelo uso dessa tecnologia.
R: Algumas das vantagens é que como ela é uma linguagem funcional, ela prioriza a utilização de funções como base da criação de seus códigos. O mercado paga bem por ter uma baixa mão de obra que utilize essa linguagem. Ela é muito flexível, ela prioriza a imutabilidade. Elixir é uma linguagem dinâmica e funcional para construir aplicativos escaláveis e de fácil manutenção. Devemos optar pelo uso dessa tecnologia em desenvolvimento web, software incorporado, aprendizado de máquina, entre outros
.

Questão 3 - [0,5 pontos] - Escreva uma função que verifique se um número é par.
```
defmodule Prova do
   def verificaPar (numero) do
      if rem(numero,2) == 0 do
      IO.puts "par"
      else
      IO.puts "não é par"
      end
   end
end
IO.inspect(Prova.verificaPar(10))
```

Questão 4 - [0,5 pontos] - Implemente uma função que calcule o dobro de cada elemento em uma lista.

```
numbers = [1,2,3,4,5]
doubled_numbers = Enum.map(numbers, &(&1 * 2))

IO.inspect(doubled_numbers)
```

Questão 5 - [0,5 ponto] - Crie uma função que retorne o último elemento de uma lista.

Questão 6 - [1 ponto] - Implemente uma função que calcule o fatorial de um número usando recursão.
```
defmodule Prova do
  def fatorial(0), do: 1
  def fatorial(n) when n > 0 do
```

```
    n * fatorial(n - 1)
  end
end

IO.puts(Prova.fatorial(5))
```

Questão 7 - [1 ponto] - Escreva uma função que aplique uma função passada como argumento a cada elemento de uma lista.

```
defmodule Prova do
  def executar(funcao, params), do: apply(funcao, params)
end

resultado1 = Prova.executar(fn (a,b,c) -> a+b+c end, [1,2,3])
resultado2 = Prova.executar(fn (c,d) -> c * d end, [5,5])

IO.puts "#{resultado1}, #{resultado2}"
```

Questão 8 - [1 ponto] - Escreva uma função que filtre os elementos de uma lista com base em uma função de filtro passada como argumento.

```
numbers = [1, 2, 3, 4, 5]
even_numbers = Enum.filter(numbers, fn n -> rem(n, 2) == 0 end)
IO.inspect(even_numbers)
```

Questão 9 - [1 ponto] - Crie uma função que gere os primeiros "n" números da sequência de Fibonacci.

```
defmodule Fibonacci do
  def fibonacci (0) do 0 end
  def fibonacci (1) do 1 end
  def fibonacci (n) do fibonacci(n-1) + fibonacci(n-2) end
end

IO.puts Fibonacci.fibonacci(10)
```

Questão 10 - [2,5 ponto] - Crie uma função que calcule a média de uma turma.
O exemplo abaixo demonstra como são passadas as notas dos alunos pertencentes a uma turma.
Você deve utilizar o método reduce para calcular a média.

```
notas_da_turma = [
  {"Alice", [9.5, 8.0, 7.5]},
  {"João", [8.0, 7.0, 6.5]},
  {"Pedro", [9, 9.5, 9.0]},
   {"Lucas", []},
```

]

Dicas: Utilize map ou flatmap para extrair todas as notas dos alunos em uma única lista