Taller Programación en elixir

Nivel: bajo

- Crea una funcion que imprime en consola la cuenta regresiva desde un número que ingrese desde consola, y al terminar imprima la palabra ignición.

Nivel:medio

- Dados los dos siguientes códigos, los cuales se encargan de simular peticiones de *n* clientes, y cuya petición tarda un tiempo *t* por cliente:

  CODIGO A:

```
get_request = fn(id) ->
        :timer.sleep(t)
        "{user:{id: #{id}}}"
end

Enum.map(1..n, &get_request.(&1))
```

  CODIGO B:

```
get_request = fn(id) ->
        :timer.sleep(t)
        "{user:{id: #{id}}}"
end
async_get_request = fn(id) ->
        spawn(fn -> IO.puts(get_request.(id)) end)
end

Enum.map(1..n, &async_get_request.(&1))
```

  Teniendo en cuenta que Enum.map funciona de manera similar al map de Python, y suponiendo que se tuviera recursos ilimitados (número de cores, memoria). ¿Cuál sería la complejidad en tiempo de cada código?

  a. CODIGO A = O(n + t); CODIGO B = O(n)
  b. CODIGO A = O(n * t); CODIGO B = O(n + t)
  c. CODIGO A = O(n + t); CODIGO B = O(t)
  d. CODIGO A = O(n * t); CODIGO B = O(t)

Nivel: medio-alto

- Escriba un programa que dado una palabra encuentre una sublista de anagramas de una lista

Nivel: alto

- Escriba un programa en Elixir que dado un grupo de personas y un conjunto de comidas, se asigne a la persona tomada de forma aleatoria, una de las comidas hasta completar todas las comidas en la lista:
  Ejemplo:
  Personas: [“Bob”, “Alice”, “Charlie”]
  Comidas: [“arroz”, “pollo”, “huevo”, “arroz”, “frijoles”, “arroz”, “huevo”]

  Salida:
  Bob toma una porción de arroz
  Charlie toma una porción de pollo
  Bob toma una porción de huevo
  Alice toma una porción de arroz
  Bob toma una porción de frijoles
  Charlie toma una porción de arroz
  Alice toma una porción de huevo

Los procesos se deben correr de manera concurrente (se pueden basar en el ejemplo del caso de uso).