Final - Algoritmos I Taller

Debés entregar el código de todos los ejercicios como un archivo adjunto de Haskell, y pegar el código correspondiente a cada ejercicio abajo de cada pregunta. El archivo se debe poder ejecutar correctamente.

Descripción del problema

Para un inventario de una reserva de animales, se necesita un sistema en Haskell para especificar el tipo Animal, que puede ser Vertebrado o Invertebrado. Estas categorías tienen diferentes características, por eso será necesario definir un tipo específico Animal y algunas funciones asociadas.

Ejercicio 1

El tipo Animal que tiene 2 constructores:

-**Vertebrado** con parámetros Nombre (sinónimo de String), Peso (sinónimo de Int), Peligrosidad (Int), éste último indica cuán peligroso es del 1 al 10 (siendo 1 no peligroso y 10 muy peligroso).

-**Invertebrado** con parámetros Nombre, TienePatas (cuyas posibilidades son Si o No un tipo enumerado), y Peligrosidad igual que en el constructor anterior.

a) Implemente los tipos Peso, Nombre, TienePatas

b) Implemente el tipo Animal.

c) Defina los siguientes animales (ejemplos):

perroCaniche es un vertebrado de nombre “Pompon”, pesa 6 kg, y su nivel de peligrosidad es 1.

ArañaPollito es un invertebrado de nombre “Pollito”, que si tiene patas y es de peligrosidad 10.

Implementación

Pegar acá el código implementado para este ejercicio. Además, todos los ejercicios deben estar en un solo archivo que se adjunta a la entrega de classroom.

```
-- Implementación del ejercicio 1
```

Ejercicio 2

Defina las siguiente funciones no recursivas sobre para cualquier tipo de Animal.

a) Escriba una función llamada `es_peligroso :: Animal -> Bool` que tome un Animal devuelva True en caso de que la peligrosidad sea mayor estricta que 5, False en caso contrario.

b) Escriba una función llamada `o cuanto_pesa:: Animal -> Int` que tome un Animal y en caso de ser un vertebrado devuelva el peso del mismo, y en caso ser invertebrado devuelva -1

Implementación

Pegar acá el código implementado para este ejercicio. Además, todos los ejercicios deben estar en un solo archivo que se adjunta a la entrega de classroom.

```
-- Implementación del ejercicio 2
```

Ejercicio 3

Definir por recursión la función

```
con_patas_peligrosos :: [Animal] -> Int -> [Nombre]
```

que dada una lista de animales y un número de peligrosidad `n` devuelve la lista de los nombres de los animales que tienen patas y peligrosidad mayor estricta que `n` que están en la lista.

Implementación

Pegar acá el código implementado para este ejercicio. Además, todos los ejercicios deben estar en un solo archivo que se adjunta a la entrega de classroom.

```
-- Implementación del ejercicio 3
```