Guardería de Mascotas

Entrega 1

Los Hermanos Malahuena administran una guardería de mascotas y nos piden que les desarrollemos un programa en Haskell que los va a ayudar en su trabajo diario.

Por lo que tendremos que implementar las siguientes funciones

1. **edadAnimal**

Esta función recibe una cantidad de años y devuelve cuantos años representan para un animal, los cuales se calculan de la siguiente manera:

- El primer año de vida vale por uno.
- Del segundo al cuarto valen dos.
- Del quinto al décimo valen tres.
- A partir del décimo valen cuatro.

Resolver este punto usando recursividad.

Ejemplos:

```
>edadAnimal 1
1

>edadAnimal 5
10
(es 10 porque es igual a 1 + 2 * 3 (los años de 2 a 4 valen
doble)+ 3 (el año 5 vale triple)
```

2. animalPachorriento

La función animalPachorriento recibe un peso en kilos y una edad en términos humanos y responde si un animal es o no pachorriento. Esto se da cuando el peso supera a la mitad de su edad animal.

Tip: vale usar la función edadAnimal como función auxiliar.

Ejemplos:

```
>animalPachorriento 3 3
True
(porque su peso supera a 2.5 que es la mitad de su edad animal)

>animalPachorriento 4 5
False
(porque 4 no supera a 5 que es la mitad de su edad animal)
```

3. sumatoriaDeExcesosDeEdad

Esta función calcula la sumatoria de las diferencias de edades entre un animal y un humano en un rango de edades determinado

Ejemplos:

```
>sumatoriaDeExcesosDeEdad 5 7
21
(es el resultado de sumar 5 + 7 + 9)
5 es la es la diferencia entre la edad humana y la animal a los 5
años humanos, porque la edad animal es 10.
A los 6 años humanos un animal tiene 13, y a los 7 tiene 16.
```

Entrega 2

La entrega deberá contener un archivo .hs con la implementación de las funciones y un archivo .txt con las pruebas realizadas en el intérprete.

En ninguno de los puntos está permitido usar recursividad.
Es obligatorio utilizar composición en alguna parte de cada uno de los puntos.

Los animalitos alojados en la guardería pueden ser representados por la función constante mascotas que devuelve una lista de animales, cada animal está representado por una tupla de cuatro elementos (nombre, especie, fechaDeNacimiento, peso). La fecha de nacimiento está representada por una tupla de tres elementos (día, mes, año).

Un conjunto posible de mascotas sería:

```
mascotas = [("Laica", "perro", (10, 10, 2008), 25.2),
        ("Twitti", "canario", (13, 12, 2010), 1.8),
        ("Sultan", "perro", (8, 3, 2009), 73.6),
        ("Michi", "gato", (1, 4, 2010), 7.5),
        ("Tifon", "delfin", (7, 2, 2001), 35.8)]
```

La función alimentos devuelve la lista de los alimentos que consumen las mascotas

```
alimentos  =
        [("perro", "polenta con pajaritos", 1.2),
        ("gato", "sushi", 0.3),
        ("delfin", "mojarritas", 2.5),
        ("canario", "alpiste", 0.02),
        ("lagarto","cordones remojados", 4.5),
        ("elefante", "mani", 56.8)]
```

Cada alimento está representado por un tupla de tres elementos (especieDeMascotaQueLoConsume, tipoDeAlimento, pesoDeLaRacionPromedio)

Para poder calcular la edad de una mascota se cuenta con la función constante anyoActual que devuelve el año en curso.
Cuando sea necesario calcular cuantos años tiene una mascota no hay que tener en cuenta el día exacto en el que nacio. Es suficiente con la cuenta (anyoActual – año en el que nació)

```
anyoActual  = 2011
```

4. tipoAlimentoPara

Dado un animal (nombre, especie, fechaNacimiento, peso), devuelve el tipo de alimento que consume.

Ej:

```
Main>tipoAlimentoPara ("Twitti", "canario", (13, 12, 2010), 1.8)
"alpiste"
```

El resultado es “alpiste” por tratarse de un canario. La información debe ser la de la lista alimentos.

5. raciones

Dada una lista de animales, devuelve una lista de las raciones necesarias para alimentar a cada animal. Cada ración está representada por una tupla de tres elementos (nombreDelAnimal, tipoDeAlimento, pesoDeLaRacion)

Ej:

```
Main> raciones mascotas
[("Laica","polenta con
pajaritos",0.554),("Twitti","alpiste",0.046),("Sultan","polenta
con
pajaritos",1.502),("Michi","sushi",0.16),("Tifon","mojarritas",0.9
76)]
```

El peso de la ración se calcula como el 2% del peso de la mascota más el 1% de su edad animal (en esta parate hay que usar la función de la primer entrega edadAnimal).

6. glotones

Dada una especie y una lista de animales devuelve una lista con los nombres de los animales glotones de la especie dada.

Ej:

```
Main> glotones "perro" mascotas
["Sultan"]
```

El resultado es la lista que contiene al nombre sultán porque de los dos perros que hay en la lista mascotas, el único glotón es Sultan.
Un animal es considerado glotón cuando el peso de su ración supera al peso de la ración promedio del tipo de comida que consume.
En el caso de los perros que comen polenta con pajaritos y el peso de la ración promedio es de 1.2, sultán la supera porque su ración pesa 1.502.
Tampoco está permitido usar recursividad en este punto.

7. agrupadosPorEspecie

Dada una lista de especies y una lista de animales devuelve una lista de tuplas cuyo primer elemento es una especie y el segundo es una lista de nombres de animales de esa especie

Ej:

```
Main> agrupadosPorEspecie ["perro", "gato", "canario",
"cocodrilo"] mascotas
[("perro",["Laica","Sultan"]),("gato",["Michi"]),("canario",["Twit
ti"]),("cocodrilo",[])]
```

Entrega 3

La entrega deberá contener un archivo .hs con la implementación de las funciones y un archivo .txt con las pruebas realizadas en el intérprete.

Se agrega la información de los costos que tiene cada tipo de alimento:

```
costosAlimentos  =
        [("polenta con pajaritos", 15.3),
        ("sushi", 23.2),
        ("mojarritas", 3.5),
        ("alpiste", 20.12),
        ("cordones remojados", 43.5),
        ("mani", 3.06)]
```

8. a) costoRacion

Dada un ración (nombre, tipoAlimento, peso) (ver punto 5). Devuelve el costo de esa ración que se calcula multiplicacando el peso de la ración por el costo del tipo de alimento

Ej:

```
Main> costoRacion("Laica","polenta con pajaritos",0.554)
8.4762
```

Es el resultado de multiplicar el peso de la ración (0.554) por el costo de la polenta con pajaritos (15.3).
No está permitido usar recursividad ni listas por comprensión.
Puede resultar útil la función:

```
find criterio = head.filter criterio
```

b) costoTotal

Dada una lista de animales, devuelve la sumatoria de los costos de las raciones para cada animal.
Ej:

```
Main> costoTotal mascotas
39.51032
```

No se puede usar recursividad en este punto.

9. ordenadosPor

Dado un criterio y una lista de animales devuelve lista de animales ordenados por el criterio

Ej:

```
Main> ordenadosPor (\x y -> peso x < peso y) mascotas
[("Twitti","canario",(13,12,2010),1.8),("Michi","gato",(1,4,2010),
7.5),("Laica","perro",(10,10,2008),25.2),("Tifon","delfin",(7,2,20
01),35.8),("Sultan","perro",(8,3,2009),73.6)]
```

El criterio de ordenamiento es una función que reicibe dos animales y devuelve True si el primer animal es anterior al segundo. En el caso del ejemplo el criterio de orden es por peso de menor a mayor.
En este punto se puede usar recursividad.

10. A) criterioGloton

Criterio glotón es un criterio de ordenamiento para usar con ordenadosPor donde el resultado son los animales ordenados por su índice de glotonería. El índice de glotonería se calcula como el peso de la porción promedio del alimento que consume el animal menos el peso de su ración (ver punto 4)

Ej:

```
Main> ordenadosPor criterioGloton mascotas
[("Tifon","delfin",(7,2,2001),35.8),("Laica","perro",(10,10,2008),
25.2),("Michi","gato",(1,4,2010),7.5),("Twitti","canario",(13,12,2
010),1.8),("Sultan","perro",(8,3,2009),73.6)]
```

El primer animalito de la lista es Tifón ya que su índice de glotonería es 0.976 – 2.5 = -1.524. 2.5 es el peso de la racion promedio de mojarritas y 0.976 es el peso de la ración que consume Tifón.

B) criterioCosto

Este criterio es para ordenar por el costo de las raciones.

Ej:

```
Main> ordenadosPor criterioCosto mascotas
[("Twitti","canario",(13,12,2010),1.8),("Tifon","delfin",(7,2,2001
),35.8),("Michi","gato",(1,4,2010),7.5),("Laica","perro",(10,10,20
08),25.2),("Sultan","perro",(8,3,2009),73.6)]
```

Para comprobar el orden por costo podemos obtener la lista de todas las raciones con sus costos:

```
Main> [(racion,costoRacion racion) | racion <- raciones
(ordenadosPor criterioCosto mascotas)]
[(("Twitti","alpiste",0.046),0.92552),(("Tifon","mojarritas",0.976
),3.416),(("Michi","sushi",0.16),3.712),(("Laica","polenta con
pajaritos",0.554),8.4762),(("Sultan","polenta con
pajaritos",1.502),22.9806)]
```

C) maximoPor / minimoPor

Usando la función ordenadoPor obtener el maximo y el minimo dado un criterio.
De esta manera se puede obtener el animal más glotón o el menos costoso

Ej:

```
Main> maximoPor criterioGloton mascotas
("Sultan","perro",(8,3,2009),73.6)

Main> minimoPor criterioCosto mascotas
("Twitti","canario",(13,12,2010),1.8)
```