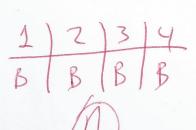
Universidad de Ecodos Aires Facultad de Ciencias Exactes y Naturales Departamento de Computación

Taller de Algebra I - Parcial

TURNO TARDE

14 de octubre de 2016



Aclaraciones

- El parcial se aprueba con al menos dos ejercicios bien resueltos.
- Programe todas las funciones en lenguaje Haskell.
- Incluya la signatura de todas las funciones que escriba.
- No está permitido alterar los tipos de datos presentados en el enunciado.

Ejercicio 1

Programe la función kesimo :: [Integer] -> Integer -> Integer, tal que kesimo xs k devuelve el elemento que está en la posición k de xs y se indefine si no existe tal elemento.

Por ejemplo:

- kesimo [1,2,4] 3 ~> 4
- kesimo [1,2,4] 7 \leadsto Indefinido

Ejercicio 2

Programe la función domina :: [Integer] -> Bool, tal que domina xs ys devuelve True si y solo si cada elemento de xs es mayor al que se encuentra en la misma posición de ys. Asuma que las listas son de igual longitud.

Por ejemplo:

- domina [1,2,3] [0,0,0] → True
- domina [2,8,3] [1,2,1] → True
- domina [2,8,3] [1,2,4] → False

Ejercicio 3

Programe la función esTipoFibonacci :: [Integer] -> Bool que, dada una lista de al menos tres elementos, devuelve True si todos los elementos a partir del tercero son la suma de los dos anteriores y False en otro caso.

Por ejemplo:

- esTipoFibonacci [3,4,7,11,18] → True
- esTipoFibonacci [3,4,6,9,14] \rightsquigarrow False

Ejercicio 4

Sean los tipos de datos:

```
data Preferencia = Carnivoro | Herbivoro deriving Show
type Especie = String
type Animal = (Especie, Preferencia)
```

Programe la función quitar :: Animal -> [Especie] -> [Especie] -> [Especie], tal que quitar animal carnivoros herbivoros devuelve la lista de carnívoros o herbívoros quitando la especie del animal, dependiendo de su Preferencia. Asuma que la primer lista solo tiene carnívoros y la segunda solo herbívoros.

Por ejemplo:

 $lue{}$ quitar ("Leon", Carnivoro) ["Leon", "Tigre"] ["Oveja"] \leadsto ["Tigre"]

domina::[Integer] -> [Integer] -> Bool

domina xs ys length ys == 0 = False x xs ys

Length Ys == 1 = dominaux xs ys

Length Ys > 1 && head xs <= head ys = False

Length ys > 1 && head xs > head ys = domina (tail xs) (tail ys)

dominaux :: [Integer] -> [Integer] -> Bool

dominaux XS yS | head XS > head YS = True

| head XS <= head YS = False.

- o me fijo solo en el largo de Ys (Length Ys) porque según el enonciado asumo que ambas listas son de igual longitud.
- 3) esTipo Fibonacci: [Integer] -> Bool.

estipofibonacci xs length xs == 3 = estipoaux xs.

length xs >3 && head xs + head (tail xs) = head (tail (tail xs)) = False length xs >3 && head xs + head (tail xs) == head (tail (tail xs)) = @

@: esTipo Fibonacci (tail xs)

estipodux :: [Integer] -> Bool

estipoaux xs | head xs + head (tail xs) = head (tail (tail xs)) = False | head xs + head (tail xs) == head (tail (tail xs)) = True.

el enunciado asi la específica (dada una lista de al menos tres elementos...).

4) data Preferencia = Carnivoro Herviboro deriving Show type Especie = String type Animal = (Especie, Preferencia) quitar :: Animal -> [Especie] -> [Especie] -> [Especie]. quitar (esp; car) w carnivoro _ = quitar esp carnivoro - herbivoro = quither esp herbivoro quitar (esp, her) quitcar: Especie -> [Especie] quiteer esp carnivoro | Length carnivoro == 0 = [] head carnivoro = zesp = tell carnivoro carnivoro /= esp = [head carnivoro]++quitcar esp (teil carnivoro) quither: Especie - [Especie] head herbivoro == 0 = []

head herbivoro == esp = tail herbivoro] ++ quither esp (tail herbivoro) quither esp herbivors · Asumo que Las especies no se cepiten en las listas