Universidad de Buenos Aires
Facultad de Ciencias Exactas y Naturales
Departamento de Computación

Taller de Álgebra I - Parcial

TURNO TARDE

14 de octubre de 2016

1	2	3	4
B	B	B	B

(A)

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` ⇝ *Indefinido*

Ejercicio 2

Programe la función `domina :: [Integer] -> [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]` ⇝ `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:

- `quitar ("Leon", Carnivoro) ["Leon", "Tigre"] ["Oveja"]` ⇝ `["Tigre"]`

1) kesimo :: [Integer] → Integer → Integer

```
kesimo xs k | k == 1 = head xs
            | otherwise = kesimo (tail xs) (k-1) .
```

2) domina :: [Integer] → [Integer] → Bool

```
domina xs ys | length ys == 0 = False
             | 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 .
```

- me fijo sólo en el largo de ys (length ys) porque según el enunciado asumo que ambas listas son de igual longitud.

3) esTipoFibonacci :: [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)) = ⊛
```

⊛ : esTipoFibonacci (tail xs)

```
estipoaux :: [Integer] → Bool
estipoaux xs | head xs + head (tail xs) /= head (tail (tail xs)) = False
             | head xs + head (tail xs) == head (tail (tail xs)) = True .
```

- no contemplé casos en los que la longitud de la lista sea menor a 3 porque el enunciado así lo especifica (dada una lista de al menos tres elementos...) .

4)

```
data Preferencia = Carnivoro | Herbivoro  deriving Show

type Especie = String

type Animal = (Especie, Preferencia)

quitar :: Animal -> [Especie] -> [Especie] -> [Especie]

quitar (esp, Carnivoro) carnivoro _ = quitcar esp carnivoro

quitar (esp, her) _ herbivoro = quither esp herbivoro

quitcar :: Especie -> [Especie]

quitcar esp carnivoro | length carnivoro == 0 = []
                      | head carnivoro == esp = tail carnivoro
                      | head carnivoro /= esp = [head carnivoro] ++ quitcar esp (tail carnivoro)

quither :: Especie -> [Especie]

quither esp herbivoro | length herbivoro == 0 = []
                      | head herbivoro == esp = tail herbivoro
                      | head herbivoro /= esp = [head herbivoro] ++ quither esp (tail herbivoro)
```

- Asumo que las especies no se repiten en las listas.