Introducción a la Programación

Primer Parcial - Turno Mañana

- El parcial se aprueba con 6 puntos
- Podrás utilizar las siguientes funciones del prelude
    - Listas: head, tail, last, init, length, elem, ++
    - Tuplas: fst, snd
    - Operaciones lógicas: &&, ||, not
    - Constructores de listas: (x:xs), []
    - Constructores de tuplas: (x,y)
- Si querés utilizar Hunit para testear tu código acá tenés un script de ejemplo.

Viva la democracia:

La elección periódica de los gobernantes es la base de los Estados Modernos. Este sistema, denominado "democracia" (término proveniente de la antigua Grecia), tiene diferentes variaciones, que incluyen diferentes formas de elección del/a máximo/a mandatario/a. Por ejemplo, en algunos países se eligen representantes en un colegio electoral (EEUU). En otros se vota a los/as miembros del parlamento (España). En nuestro país elegimos de forma directa la fórmula presidencial (Presidente/a y Vicepresidente/a) cada 4 años.

A continuación presentamos una serie de ejercicios que tienen como objetivo implementar funciones para sistema de escrutinio de una elección presidencial. Leer las descripciones y especificaciones e implementar las funciones requeridas en Haskell, utilizado sólamente las herramientas vistas en clase.

Las fórmulas presidenciales serán representadas por tuplas (String x String), donde la primera componente será el nombre del candidato a presidente, y la segunda componente será el nombre del candidato a vicepresidente.

En los problemas en los cuales se reciban como parámetro secuencias de fórmulas y votos, cada posición de la lista *votos* representará la cantidad de votos obtenidos por la fórmula del parámetro *formulas* en esa misma posición. Por ejemplo, si la lista de fórmulas es [("Juan Pérez","Susana García"), ("María Montero","Pablo Moreno")] y la lista de votos fuera [34, 56], eso indicaría que la fórmula encabezada por María Montero obtuvo 56 votos, y la lista encabezada por Juan Pérez obtuvo 34 votos.

---

1) Votos en Blanco [1 punto]

problema votosEnBlanco (*formulas*: seq(String x String),*votos*:seq< Z >, *cantTotalVotos*: Z) : Z {
  requiere: {formulasValidas(*formulas*)}
  requiere: {|*formulas*| = |*votos*|}
  requiere: {Todos los elementos de *votos* son mayores o iguales a 0}
  requiere: {La suma de todos los elementos de *votos* es menor o igual a *cantTotalVotos*}
  asegura: {*res* es la cantidad de votos emitidos que no correspondieron a niguna de las fórmulas que se presentaron}
}

---

2) Formulas Válidas [3 puntos]

problema formulasValidas (*formulas*: seq⟨String x String⟩) : Bool {
  requiere: {True}
  asegura: {(*res* = true) <=> *formulas* no contiene nombres repetidos, es decir que cada candidato está en una única fórmula (no se puede ser candidato a presidente y a vicepresidente ni en la misma fórmula ni en fórmulas distintas)}
}

---

3) Porcentaje de Votos [3 puntos]

problema porcentajeDeVotos (*presidente*: String, *formulas*: seq⟨String x String⟩,*votos*:seq< Z >) : R {
  requiere: {La primera componente de algún elemento de *formulas* es *presidente*}
  requiere: {formulasValidas(*formulas*)}
  requiere: {|*formulas*| = |*votos*|}
  requiere: {Todos los elementos de *votos* son mayores o iguales a 0}
  requiere: {Hay al menos un elemento de *votos* que es mayor que estricto que 0}
  asegura: {*res* es el porcentaje de votos que obtuvo la fórmula encabezada por *presidente* sobre el total de votos afirmativos}
}

Para resolver este ejercicio pueden utilizar la siguiente función que devuelve como Float la división entre dos números de tipo Int:

```
division :: Int -> Int -> Float
division a b = (fromIntegral a) / (fromIntegral b)
```

---

4) Próximo Presidente [3 puntos]

problema proximoPresidente (*formulas*: seq(String x String), *votos*:seq< Z >) : String {
  requiere: {formulasValidas(*formulas*)}
  requiere: {|*formulas*| = |*votos*|}
  requiere: {Todos los elementos de *votos* son mayores o iguales a 0}
  requiere: {Hay al menos un elemento de *votos* mayores estricto a 0}
  requiere: {|*formulas*| > 0}
  asegura: {*res* es el candidato a presidente de *formulas* más votado de acuerdo a los votos contabilizados en *votos*}
}

Completa aca el codigo Haskell:

A continuación te dejamos una estructura básica para resolver los ejercicios. Este código no pretende resolver ningun caso de los ejercicios planteados, es sólo una plantilla.

```
module Solucion where

-- Ejercicio 1
votosEnBlanco :: [(String, String)] -> [Int] -> Int  -> Int
votosEnBlanco _ _ _ = 0

-- Ejercicio 2
formulasValidas :: [(String, String)] -> Bool
formulasValidas _ = True

-- Ejercicio 3
porcentajeDeVotos :: String -> [(String, String)] -> [Int] -> Float
porcentajeDeVotos _ _ _ = 0.0

-- Ejercicio 4
proximoPresidente :: [(String, String)] -> [Int] -> String
proximoPresidente _ _ = ""
```

Enviar