Resultado parcial Haskell IP/Algo1

Parcial - tema2

Nota: 9.83 / 10.0 (APROBADO)

```
puntaje ej1: 0.9
puntaje ej2: 2.93
puntaje ej3: 3
puntaje ej4: 3
```

Datos del alumno

Nombre: mateos tomas

Enunciado

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 [/static/hunit_example.html]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) Porcentaje de Votos Afirmativos [1 punto]

problema porcentajeDeVotosAfirmativos (*formulas*: seq⟨String x String⟩,*votos*:seq< Z >, *cantTotalVotos*: Z) : R {
  requiere: {¬formulasInvalidas(*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 el porcentaje de votos no blancos (es decir, asociados a alguna de las fórmulas) sobre el total de votos emitidos}
}

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)
```

---

2) Formulas Inválidas [3 puntos]

problema formulasInvalidas (*formulas*: seq⟨String x String⟩) : Bool {
  requiere: {True}
  asegura: {(*res* = true) <=> *formulas* contiene un candidato se propone para presidente y vicepresidente en la misma fórmula; o algún candidato se postula para presidente o vice en más de una fórmula }

---

3) Porcentaje de Votos [3 puntos]

problema porcentajeDeVotos (*vice*: String, *formulas*: seq⟨String x String⟩,*votos*:seq< Z >) : R {
  requiere: {La segunda componente de algún elemento de *formulas* es *vice*}
  requiere: {¬formulasInvalidas(*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}
  asegura: {*res* es el porcentaje de votos que obtuvo *vice* sobre el total de votos afirmativos}
}

Para resolver este ejercicio pueden utilizar la función `division` presentada en el Ejercicio 1.

---

4) Menos Votado [3 puntos]

problema menosVotado (*formulas*: seq⟨String x String⟩, *votos*:seq< Z >) : String {
  requiere: {¬formulasInvalidas(*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* menos votado de acuerdo a los votos contabilizados en *votos*}
}

Solucion entregada por el alumno

```
--Problema 1: Porcentaje de votos afirmativos
porcentajeDeVotosAfirmativos :: [Int] -> Int -> Float
porcentajeDeVotosAfirmativos vr votos = (division (sumaVotos vr) votos) *100

sumaVotos :: [Int] -> Int
sumaVotos [] = 0
sumaVotos (v:vs) = v + sumaVotos vs

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

--lo que hace la funcion es sumar los votos de toda la lista de votos, y
dividirlo por el voto total. multiplicado por 100 para que muestre 90 en vez de
0.9

-- la funcion tendria que ingresar una tupla de candidatos, los votos recibidos
y los votos totales, y devolver el porcentaje de votos.
-- ejemplo: [34, 56] 100 devuelve 90, ya que 100 son los votos totales y 90 los
votos recibidos
-- no tiene sentido para este problema ingresar la lista de tuplas, ya que el
calculo se realiza solo con los votos.

--Problema 2: Formulas invalidas
formulasInvalidas :: [(String,String)] -> Bool
formulasInvalidas [] = True
formulasInvalidas ((a,b):xs) = repetidos "" (separarEnLista ((a,b):xs))

separarEnLista :: [(String,String)] -> [String]
separarEnLista [] = []
separarEnLista ((a,b):xs) = a:b : separarEnLista xs

repetidos :: (Eq a) => a -> [a] -> Bool
repetidos _ [] = False
repetidos a (x:xs) | a == x = True
                   | otherwise = repetidos a (xs) || repetidos x xs

--lo que hace la funcion es checkear que no haya ningun nombre repetido en las
tuplas.
-- para ello separa la lista de tuplas en una lista comun y mira que no haya
dos personas repetidas.

--las formulas son invalidas si hay dos tuplas con el mismo nombre o una tupla
con la misma persona en los dos espacios.
-- formulasInvalidas [("jorge","tomas"),("tomas","martin")] devuelve True ya
que tomas esta en dos tuplas
-- formulasInvalidas [("tomas","tomas"),("jorge","martin")] devuelve True ya
que tomas ocupa los dos lugares de una tupla, jorge martin es una formula
valida.
-- formulasInvalidas [("tomas","nacho"),("jorge","martin")] devuelve False ya
que no hay un candidato en dos lugares distintos.
```

```
--Problema 3: Porcentaje de votos
porcentajeDeVotos :: String -> [(String,String)] -> [Int] -> Float
porcentajeDeVotos vice candidatos votos = (division (devolverEspacioX votos
(buscarTupla vice candidatos)) (sumaVotos votos)) * 100

buscarTupla :: String -> [(String,String)] -> Int
buscarTupla candidato ((a,b):xs) | candidato == b || candidato == a = 0 --la
funcion busca en que tupla esta el vicepresidente ingresado --no cancele que
busque presidente porque reuse la funcion para el punto 4
                                 | otherwise = 1 + buscarTupla candidato (xs)

devolverEspacioX :: [Int] -> Int -> Int
devolverEspacioX (x:xs) espacio | espacio == 0 = x --esta funcion devuelve el
espacio de la lista pedido, si se ingresa [2,3,4,6] 2 devuelve el 4.
                                | otherwise =devolverEspacioX xs (espacio-1)

--la funcion completa busca donde esta el vicepresidente en la lista de tuplas,
-- devuelve el numero de espacio en que esta sus votos y luego calcula el
porcentaje de votos que recibio

--devuelve el porcentaje de votos de un vicepresidente ingresado.
-- por ejemplo tomas [("martin","tomas"),("jorge","lautaro")] [40,60] deberia
devolver 40 ya que la formula martin tomas obtuvo 40 votos de 100

--Problema 4: Menos votado.
menosVotado :: [(String,String)]->[Int]->String
menosVotado [(a,b)] [x] = a
menosVotado ((p1,v1):(p2,v2):cs) (x1:x2:xs) | numeroDeVotos p1 ((p1,v1):
(p2,v2):cs) (x1:x2:xs) <= numeroDeVotos p2 ((p1,v1):(p2,v2):cs) (x1:x2:xs) =
menosVotado ((p1,v1):cs) (x1:xs)
                                            | otherwise = menosVotado
((p2,v2):cs) (x2:xs)

numeroDeVotos :: String->[(String,String)] -> [Int] -> Int
numeroDeVotos p ((p1,v1):xs) votos = devolverEspacioX votos (buscarTupla p
((p1,v1):xs))
--reusa las funciones del punto 3, busca la tupla donde esta el presidente y
devuelve los votos que tiene.

--dada una lista de candidatos y una lista de votos, devuelve el candidato a
presidente menos votado.
--si votos de c1 <= c2 entonces quiero borrar c2 y comparar c1 con c3
--si votos de c1 >= c2 entonces c2 es menor, quiero comparar c2 con c3
```

Resultado de la compilacion

```
[1 of 1] Compiling Main             ( submissions/mateos_tomas_645_23
/submission.hs.main.hs, submissions/mateos_tomas_645_23/submission.hs.main.o )
Linking submissions/mateos_tomas_645_23/submission.hs.main ...
```

Cambios realizados al codigo

La respuesta original entregada tenia errores en la compilacion, por lo que los docentes realizaron cambios a la misma. Estos fueron los cambios realizados:

```
diff submissions/mateos_tomas_645_23/submission.hs
submissions/mateos_tomas_645_23/submission.hs.fixed.hs

0a1,6
> module Solucion where {- CORRECTOR JMF: AUSENCIA DE ESTA LINEA -}
>
> {- RESUMEN CORRECCIONES PARA LOGRAR EL COMPIALADO:
> - SOLO AGREGUE UN PARAMETRO QUE FALTA EN LA PRIMERA FUNCIÓN, NADA MÁS.
>  -}
>
2,3c8,11
< porcentajeDeVotosAfirmativos :: [Int] -> Int -> Float
< porcentajeDeVotosAfirmativos vr votos = (division (sumaVotos vr) votos) *100
---
> -- porcentajeDeVotosAfirmativos :: [Int] -> Int -> Float {- CORRECOTR JMF:
FALTA UN PARAMETRO -}
> -- porcentajeDeVotosAfirmativos vr votos = (division (sumaVotos vr) votos)
*100
> porcentajeDeVotosAfirmativos :: [(String, String)] -> [Int] -> Int -> Float
{- CORRECOTR JMF: FALTA UN PARAMETRO -}
> porcentajeDeVotosAfirmativos _ vr votos = (division (sumaVotos vr) votos)
*100
```

Resultado de la compilacion corregida

```
[1 of 1] Compiling Solucion         ( submissions/mateos_tomas_645_23
/submission.hs.fixed.hs.main.hs, submissions/mateos_tomas_645_23
/submission.hs.fixed.hs.main.o )
```

Ejecucion de los tests

Tema2-test-ej1.hs.compilacion.out

Puntaje del ej: 0.9 / 1

```
### Failure in: 0:test sintaxis
Tema2-test-ej1.hs:8
expected: True
 but got: False

Cases: 10  Tried: 10  Errors: 0  Failures: 1
```

Tema2-test-ej2.hs.compilacion.out

Puntaje del ej: 2.93 / 3

```
### Failure in: 1: formulas vacias
Tema2-test-ej2.hs:9
expected: False
 but got: True

Cases: 49  Tried: 49  Errors: 0  Failures: 1
```

Tema2-test-ej3.hs.compilacion.out

Puntaje del ej: 3 / 3

```
Cases: 18  Tried: 18  Errors: 0  Failures: 0
```

Tema2-test-ej4.hs.compilacion.out

Puntaje del ej: 3 / 3

```
Cases: 18  Tried: 18  Errors: 0  Failures: 0
```

FIN