Recuperatorio Haskell - Tema 1

Importante

Template de funciones a implementar <u>aca</u> Lista de funciones permitidas <u>aca</u> Ejemplo de hunit <u>aca</u>

Enunciado

Una **sopa de números** es un juego que consiste en descubrir propiedades de un tablero de dimensiones $n \times m$ con $n \times m \times m \times m \times m \times m \times m$, en los que en cada posición hay un número entero positivo. Cada posición se identifica con una dupla (i,j) en el cual la primera componente corresponde a una fila y la segunda a una columna. A modo de ejemplo, la siguiente figura muestra un tablero de 5×4 en el que el número 13 aparece en la posición (1,1) y el número 5 aparece en la posición (4,3). Notar que tanto la numeración de las filas como la de las columnas comienzan en 1.

13	12	6	4
1	1	32	25
9	2	14	7
7	3	5	16
27	2	8	18

Un camino en un tablero está dado por una secuencia de posiciones adyacentes en la que solo es posible desplazarse desde una posición dada hacia la posición de su derecha o hacia la que se encuentra debajo. En otras palabras, un camino de longitud / en un tablero se define como una secuencia con / posiciones, ordenadas de manera tal que el elemento *i*-ésimo es la posición resultante de haberse movido hacia la derecha o hacia abajo desde la posición (*i*-1)-ésima. Siguiendo con el ejemplo, a continuación puede observarse un camino de longitud 5 que representa la sucesión Fibonacci y que empieza en la posición (2,1) y termina en (4,3) del tablero.

13	12	6	4
1	1	32	25
9	2	14	7
7	3	5	16
27	2	8	18

Para manipular las sopas de números en Haskell vamos a representar el tablero como una lista de filas de igual longitud. A su vez, cada fila vamos a representarla como una lista de enteros positivos. Las posiciones vamos a representarlas con tuplas de dos números enteros positivos y un camino va a estar dado por una lista de posiciones.

Para implementar esta sopa de números nos enviaron las siguientes especificaciones y nos pidieron que hagamos el desarrollo enteramente en Haskell, utilizando los tipos requeridos y solamente las funciones que se ven en la materia Introducción a la Programación / Algoritmos y Estructuras de Datos I (FCEyN-UBA).

Asumimos los siguientes renombres de tipos de datos en las especificaciones de los ejercicios:

- Fila = $seq\langle Z\rangle$
- Tablero = seq<Fila>

```
- Posicion = Z x Z -- Observación: las posiciones son: (fila, columna)
  - Camino = seq<Posicion>
1. Ejercicio 1 (2 puntos)
  problema maximo (t. Tablero) : Z {
   requiere: {El tablero t es un tablero bien formado, es decir, la longitud de todas las filas es la misma, y
  tienen al menos un elemento}
   requiere: {Existe al menos una columna en el tablero t }
   requiere: {El tablero t no es vacío, todos los números del tablero son positivos, mayor estricto a 0}
   asegura: { res es igual al número más grande del tablero t}
2. Ejercicio 2 (2 puntos)
  problema masRepetido (t: Tablero) : Z {
   requiere: {El tablero t es un tablero bien formado, es decir, la longitud de todas las filas es la misma, y
  tienen al menos un elemento}
   requiere: {Existe al menos una columna en el tablero t }
   requiere: {El tablero t no es vacío, todos los números del tablero son positivos, mayor estricto a 0}
   asegura: {res es igual al número que más veces aparece en un tablero t. Si hay empate devuelve cualquiera
  de ellos}
3. Ejercicio 3 (2 puntos)
  problema valoresDeCamino (t. Tablero, c. Camino) : seq<Z> {
   requiere: {El tablero t es un tablero bien formado, es decir, la longitud de todas las filas es la misma, y
  tienen al menos un elemento}
   requiere: {Existe al menos una columna en el tablero t }
   requiere: {El tablero t no es vacío, todos los números del tablero son positivos, mayor estricto a 0}
   requiere: {El camino c es un camino válido, es decir, secuencia de posiciones adyacentes en la que solo es
  posible desplazarse hacia la posición de la derecha o hacia abajo y todas las posiciones están dentro de los
  limites del tablero t}
   asegura: {res es igual a la secuencia de números que están en el camino c, ordenados de la misma forma
  que aparecen las posiciones correspondientes en el camino.}
  }
4. Ejercicio 4 (2 puntos)
  problema esCaminoFibo (s: seq\langle Z \rangle, i: Z) : Bool {
   requiere: {La secuencia de números s es no vacía y está compuesta por números positivos (mayor estricto
  que 0) que representan los valores de un camino en un tablero}
   requiere: \{i >= 0\}
   asegura: {res = true <=> los valores de s son la sucesión de Fibonacci inicializada con el número pasado
  como parámetro i}
  Notas:
```

En este ejercicio se pasa una secuencia de valores en lugar de un tablero y un camino para no generar dependencia con el ejercicio anterior.

Recordemos que la sucesión de Fibonacci está definida con la siguiente función recursiva:

```
f(0) = 0

f(1) = 1

f(n) = f(n-1) + f(n-2) con n>1
```

5. Ejercicio 5 (2 puntos)				
	oción correcta. Haskell es un lenguaje de programación con evaluación: que una expresión solo se evalúa cuando le toca, yendo de izquierda a derecha en el			
_	rque una expresión se evalúa tan pronto como se vincula a una variable o se pasa			
O Perezosa (o <i>lazy</i>) porque una expresión solo se evalúa cuando se necesita su valor.				
Adjunta el archivo con t Solo se puede adjuntar 1 ar deben ser entregados.	chivo de extensión .hs. En caso de haber desarrollado tests propios, no			
Seleccionar archivo a enviar: Choose File No file chosen				
	Enviar			

En el ejemplo del tablero y del camino (verde claro) que figuran más arriba tenemos que esCaminoFibo [1,1,2,3,5] 1 reduce a *True*.