Universidad Central de Venezuela
Facultad de Ciencias
Escuela de Computación
Lenguajes de Programación (6204)
Semestre U-2018

Proyecto #1 - Haskell
Spelunky y Spelucao

Spelunky el minero desea explorar una mina en busca de tesoros. Desafortunadamente tiene un adversario llamado Spelucao que también desea explorar una mina cercana.

Dado que Spelunky tiene experiencia en el campo, teme que ambas minas se conecten y que Spelucao se quede con parte del botín que él desea recolectar. Para salir de dudas consiguió el mapa de exploración de su adversario, y te contrató a ti para que determinaras si sus dudas son correctas. En caso de que lo sean, debes decir qué camino tomar dentro de las minas para llegar a la base de Spelucao, cómo sabotear y además qué zonas debe explotar para separarlas.

Se sabe que la mina está formada por zonas y túneles que las interconectan, aunque puede que en el mapa se registren zonas a las cuales todavía no se sabe cómo llegar.

Nota:
La base de cada campamento minero es el primer elemento de la lista.

Dada toda esta información, usted debe implementar las siguientes funciones:

Las zonas están identificadas con un entero

```
data Zona = ZonaC Int
```

Los túneles unen 2 zonas y tienen un entero que representa qué tan rápido se puede recorrer

```
data Tunel = TunelC Zona Zona Int
data Mapa =  MapaC [Zona] [Tunel]
```

```
estaSpelunkyEnPeligro :: Mapa -> Mapa -> Bool
```

Esta función es necesaria para determinar si las minas están conectadas.
Entrada: el mapa de Spelunky y el mapa del adversario
Salida: ¿están conectadas las minas?

```
prendeElGPS :: Mapa-> Mapa -> [Tunel]
```

Para sabotear a Spelucao necesitamos saber cómo llegar a su base desde la nuestra.
Entrada: el mapa de Spelunky y el mapa del adversario

Salida: lista de túneles que debemos recorrer para llegar de una base a la otra de la manera más rápida, si no hay, retornar una lista vacía

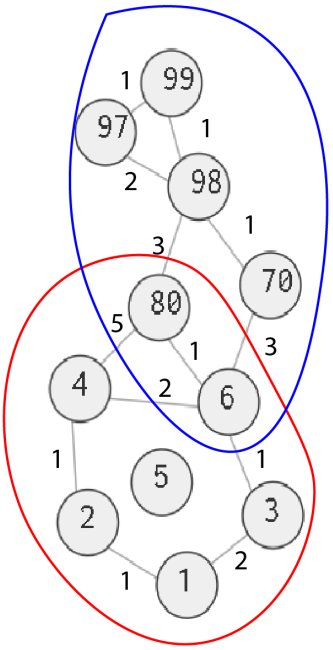
`eliminarMarcas :: Mapa -> Mapa -> (Mapa, Mapa)`
Cuando escapemos de sabotear a Spelucao necesitamos cubrir nuestros pasos explotando aquellas zonas que interconecten las minas.
Entrada: el mapa de Spelunky y el mapa del adversario
Salida: ambos mapas después de explotar todo

Ejemplo

Dados los mapas:

```
let mapaSpelunky = MapaC [ ZonaC 1, ZonaC 2, ZonaC 3, ZonaC 4,
ZonaC 5, ZonaC 6, ZonaC 80] [ TunelC (ZonaC 1) (ZonaC 2) 1, TunelC
(ZonaC 2) (ZonaC 4) 1, TunelC (ZonaC 4) (ZonaC 80) 5, TunelC
(ZonaC 4) (ZonaC 6) 2, TunelC (ZonaC 1) (ZonaC 3) 2, TunelC (ZonaC
3) (ZonaC 6) 1, TunelC (ZonaC 6) (ZonaC 80) 1 ]
```

```
let mapaSpelucao = MapaC [ ZonaC 80, ZonaC 6, ZonaC 98, ZonaC 97,
ZonaC 99, ZonaC 70 ] [ TunelC (ZonaC 99) (ZonaC 98) 1, TunelC
(ZonaC 99) (ZonaC 97) 1, TunelC (ZonaC 97) (ZonaC 98) 2, TunelC
(ZonaC 98) (ZonaC 80) 3, TunelC (ZonaC 98) (ZonaC 70) 1, TunelC
(ZonaC 70) (ZonaC 6) 3 ]
```



Llamada:

```
estaSpelunkyEnPeligro mapaSpelunky mapaSpelucao
```

Salida:

```
True
```

En este caso, notamos que las zonas 80 y 6 están en común en ambos mapas, por lo que las minas están conectadas

Llamada:

```
prendeElGPS mapaSpelunky mapaSpelucao
```

Salida:

```
[ TunelC (ZonaC 1) (ZonaC 3) 2, TunelC (ZonaC 3) (ZonaC 6) 1, TunelC (ZonaC 6) (ZonaC 80) 1, TunelC (ZonaC 80) (ZonaC 98) 3, TunelC (ZonaC 98) (ZonaC 99) 1 ]
```

Llamada:

```
Eliminar_Marcas mapaSpelunky mapaSpelucao
```

Salida:

```
(MapaC [ ZonaC 1, ZonaC 2, ZonaC 3, ZonaC 4, ZonaC 5 ] [ TunelC (ZonaC 1) (ZonaC 2) 1, TunelC (ZonaC 2) (ZonaC 4) 1, TunelC (ZonaC 1) (ZonaC 3) 2 ], MapaC [ ZonaC 98, ZonaC 97, ZonaC 99, ZonaC 70 ] [ TunelC (ZonaC 99) (ZonaC 98) 1, TunelC (ZonaC 99) (ZonaC 97) 1, TunelC (ZonaC 97) (ZonaC 98) 2, TunelC (ZonaC 98) (ZonaC 70) 1])
```

Consideraciones

- El intérprete a utilizar para la corrección del proyecto será Glasgow Haskell Compiler (GHC) en su versión 7.10.3 que puede ser descargado desde https://www.haskell.org/ghc/
- No se permite el uso de bibliotecas para el manejo de vectores y matrices en Haskell.
- La entrega del proyecto será el lunes 19/11/2018.
- El resto de las consideraciones se encuentran en el documento “Condiciones de Entrega”.

GDLP, Octubre 2018