Telco

Es una compañía que se dedica a comunicar las ciudades que se suscriben a su servicio.
Primero las ingresa al mapa de la región.
Luego establece vínculos entre ellas de cierta calidad y capacidad.
Finalmente establece canales que conectan distintas ciudades ocupando una unidad de capacidad por cada enlace recorrido.

Para sostener este modelo se cuenta con las siguientes entidades:

```
module Point ( Point, newP, difP)
       where

data Point = Poi Int Int deriving (Eq, Show)

newP :: Int -> Int -> Point
difP :: Point -> Point -> Float -- distancia absoluta
```

```
module City ( City, newC, nameC, distanceC )
       where

data City = Cit String Point deriving (Eq, Show)

newC :: String -> Point -> City
nameC :: City -> String
distanceC :: City -> City -> Float
```

{ proyecta las distancia que se obtiene de las coordenadas de cada ciudad }

```
module Quality ( Quality, newQ, capacityQ, delayQ )
       where

data Quality = Qua String Int Float deriving (Eq, Show)

newQ :: String -> Int -> Float -> Quality
capacityQ :: Quality -> Int -- cuantos túneles puede tolerar esta conexión
delayQ :: Quality -> Float -- la demora por unidad de distancia que sucede en las
conexiones de este canal
```

{ Un Link es el medio físico que une dos ciudades. }

```
module Link ( Link, newL, linksL, connectsL, capacityL, delayL )
       where

data Link = Lin City City Quality deriving (Eq, Show)
```

```
newL :: City -> City -> Quality -> Link -- genera un link entre dos ciudades distintas
connectsL :: City -> Link -> Bool -- indica si esta ciudad es parte de este link
linksL :: City -> City -> Link -> Bool -- indica si estas dos ciudades distintas están conectadas mediante este link
```

{ Un Link es naturalmente bidireccional, Si las ciudades A y B están enlazadas por un link `li`, `linksL A B li` y `linksL B A li` es true }

```
capacityL :: Link -> Int
delayL :: Link -> Float -- la demora que sufre una conexión en este canal
```

{ esta demora es en unidades de tiempo }
{ La demora de un link es en tiempo, segundos, milisegundos, etc.
La demora de la calidad de un enlace es en velocidad, por ejemplo km/segundo.
No importan las unidades, sí la relación entre los valores }

{ Un `Tunel` es la conexión lógica que podemos establecer entre dos puntos, también es bidireccional }

```
module Tunel ( Tunel, newT, connectsT, usesT, delayT )
       where

data Tunel = Tun [Link] deriving (Eq, Show)

newT :: [Link] -> Tunel
```

{ El newT recibe una lista de links ordenada y válida, donde los extremos son los extremos del túnel. (Que sea válida significa que no hace falta revisarla) }

```
connectsT :: City -> City -> Tunel -> Bool -- indica si éste túnel conecta estas dos ciudades distintas
```

{ dadas dos ciudades esta función da si si las ciudades son los extremos del túnel }

```
usesT :: Link -> Tunel -> Bool -- indica si este túnel atraviesa ese link
```

{ Un túnel recorre una serie de uno o más links, esta función indica si el link consultado es parte de esa serie }

```
delayT :: Tunel -> Float -- la demora que sufre una conexión en este túnel
```

{ esta demora es en unidades de tiempo, cuanto demora la información en recorrer el túnel }

```
module Region ( Region, newR, foundR, linkR, tunelR, connectedR, linkedR, delayR, availableCapacityForR )
       where

data Region = Reg [City] [Link] [Tunel] deriving (Show)
newR :: Region
foundR :: Region -> City -> Region -- agrega una nueva ciudad a la región
```

{ Hay decisiones que tomar! }

```
linkR :: Region -> City -> City -> Quality -> Region -- enlaza dos ciudades de la región con un enlace de la calidad indicada
```

{ Hay decisiones que tomar! }

```
tunelR :: Region -> [ City ] -> Region -- genera una comunicación entre dos ciudades distintas de la región
```

{ la lista de ciudades indica el camino ordenado de enlaces que debe tomar el túnel de inicio a fin }

{ Hay decisiones que tomar! }

```
connectedR :: Region -> City -> City -> Bool -- indica si estas dos ciudades están conectadas por un túnel
```

{ Dice si existe un túnel en la región que tenga estas ciudades por origen o destino }

```
linkedR :: Region -> City -> City -> Bool -- indica si estas dos ciudades están enlazadas
```

{ Dice si existe un enlace en la región que entre estas dos ciudades }

```
delayR :: Region -> City -> City -> Float -- dadas dos ciudades conectadas, indica la demora
```

{ Hay decisiones que tomar! }

```
availableCapacityForR :: Region -> City -> City -> Int -- indica la capacidad disponible entre dos ciudades
```

{ Teniendo en cuenta la capacidad que los túneles existentes ocupan }

{ La conexión sólo se da a través de un túnel, y sólo se conectan los extremos }

{ Cada vez que se refiere a 'conectadas', necesariamente se refiere a un túnel }

Aclaraciones Generales

- Cada Módulo debe ir en un archivo separado con nombre similar al data definido.
- Luego deben agregarse los imports correspondientes en cada módulo.
- La interfaz de cada módulo (lo que está entre paréntesis en la línea `module …`) no puede cambiar en ningún caso.