DUAL NET

Se trabajará con el siguiente TAD:

```
data DualNet = DN (Switch Cliente)    -- Indica todas las conexiones en curso
                  (Map Cliente Ruta)  -- Asocia cada cliente con su ruta`

data Cliente  = String
data Terminal = Boca1 | Boca2 deriving Eq
data Ruta     = [Terminal]
```

Enunciados
Definir invariantes de representacion para la estructura dada.
Implementar las siguientes funciones de interfaz para Dualnet justificanco los costos.
emptyDM :: Dualnet a - Costo O(1) - Describe una red sin conexiones.

cantidadDeClientesConectados :: Dualnet a -> Int - Costo O(1)

estaDisponible :: Ruta -> Dualnet a -> Bool - Costo O(L+Log C) - Dadas una ruta y una red indica si la ruta está disponible.

pinPorCliente :: Dualnet a -> Heap (Int, Cliente) - Costo O(L + C log C) - Dada una red retorna una Heap con todos los pares (Longitud de la ruta, Cliente) para ordenar la longitud de la ruta de mayor a menor.

Parte 2 - Switch
Se trabajará con el siguiente TAD:

```
data Switch a = Terminal | Conexion (RedPrivada a) (Switch a) (Switch a)

data RedPrivada a = Disponible | Conexion a
```

La unica forma en que un Switch tenga al menos una de sus conexiones en terminal.

Enunciados
Definir invariantes de representacion para la estructura dada.

Implementar las siguientes funciones de interfaz de para Switch justificanco los costos.

newSW :: Switch a - Costo: 0(1) - Descirbir un Switch nuevo sin conexiones.

conectar :: Ruta -> a -> Switch a -> Switch a - Costo: 0(r) - Dada una ruta de conexion y un Switch describe el Switch resultante de agregarle la nueva conexión al dato con la ruta dada.

desconectar :: Ruta -> Switch a -> Switch a - Costo: O(r) donde r la longitud de la lista - Dada una ruta de conexion y un Switch , describe el Switch resultante de eliminar la conexion de la ruta dada. Suponer que la ruta esta siendo utilizada en el Switch dado.

disponiblesADistancia :: Switch a -> Int -> [Ruta] - Costo: O(2^N) - Dado un Switch y un numero de ruta, describe la lista de todas las rutas disponibles en la ruta dada.

Anexo de interfaces
Map
emptyM -> Map k v O(1)
lookupM -> k -> Map k v -> Maybe v O(Log N)
assocM -> k -> v -> Map k v -> Map k v O(Log N)
deleteM -> k -> Map k v -> Map k v O(Log N)
keys -> Map k v -> [k] O(N)
sizeM -> Map k v -> Int O(1)