Existen los pokemones, que tienen un nombre (por ejemplo Charmander, Gyarados, Tauros, Carpinchos) y un tipo (planta, agua o fuego, por ahora) y se sabe que los pokemones pueden ganar contra otros según su tipo. Planta le gana a agua, agua a fuego y fuego a planta.

Resolver los siguientes puntos en Haskell escribiendo los tipos de todas las funciones que hagan. No hace falta hacer tests.

1) Dado el código inicial, se quiere conocer a qué pokemones les puede ganar un pokemon dado (es decir, a cuáles aventaja por su tipo). Pensar: ¿Qué cosas necesito recibir? ¿y qué devolver?

2) Teniendo un pokemon, se quiere conocer a cuántos puede ganarle de una lista de pokemones.

3) Conocer el pokemon que a más pokemones les puede ganar de una lista.

   Charmander (de fuego) es **el más picante** porque le puede ganar a dos pokemones, a *bulbasaur* (planta) y a *oddish* (planta) . El resto sólo le puede ganar a 1: *bulbasaur* sólo le puede ganar a *squirtle* (agua)*, squirtle* sólo le puede ganar a *charmander*, y *oddish* sólo a *squirtle*.

4) Se sabe que un destino a donde puede pelear un pokemon puede ser un gimnasio o una liga. Los gimnasios son consecutivos (se sabe cuál es el siguiente de cada uno) y al final de un camino siempre hay una liga. Por ahora sólo nos interesan los pokemones contrincantes que existen en una liga.
   Dada la siguiente definición

```
data Destino = UnGimnasio {nombre:: String, siguiente:: Destino}
             | UnaLiga {contrincantes:: [Pokemon] }
```

   Se desea saber si un pokemon está al horno en un destino. En un gimnasio, un pokemon siempre está al horno, y en una liga, está al horno cuando todos los contrincantes pueden ganarle.

5) Saber si puedo viajar de un destino al otro.
   Consideraciones a tener en cuenta:
   - Desde una Liga **no** puedo viajar a otro destino.
   - Desde unGimnasio puedo viajar a miDestino si miDestino se encuentra entre los siguientes destinos de unGimnasio. Es decir, miDestino debe estar en el camino a seguir de unGimnasio.

Por ejemplo: tenemos el **gymRoca**, al que le sigue **gymAgua**, al que le sigue **gymElectrico** y termina en **ligaKanto**. Y después tenemos el **gymFuego** al que le sigue el **gymPlanta** y termina en **ligaGali.** Entonces:

```
Main> puedoViajar ligaKanto gymRoca
False  -- desde una liga no puedo viajar a ningún destino

Main> puedoViajar gymRoca gymAgua
True   -- el gymAgua es el destino siguiente inmediato del gymRoca

Main> puedoViajar gymRoca ligaKanto
True   -- desde el gymRoca eventualmente llego a la ligaKanto

Main> puedoViajar gymRoca gymPlanta
False  -- desde el gymRoca no puedo llegar al gymPlanta porque no está en el camino.

Main> puedoViajar gymRoca ligaGali
False  -- ídem explicación anterior
```