# Trabajo Práctico

Redes Sociales

Taller de Álgebra
Primer cuatrimestre 2019

Fecha límite de entrega:
viernes 21 de junio hasta las **23:59** hs.
Coloquio:
miércoles 26 y viernes 28 de junio.

## Introducción

El objetivo de este Trabajo Práctico es aplicar los conceptos matemáticos vistos en la materia para programar un ejemplo de red social utilizando Haskell.

En nuestro ejemplo, los usuarios solamente pueden realizar las siguientes **acciones**:

- Relacionarse entre sí, es decir, *ser amigos*.
- Postear publicaciones de texto.
- Dar *me gusta* a las publicaciones de sus amigos.

La **red social** se define a partir de los **usuarios**, las **publicaciones** de cada uno y las **relaciones de amistad** entre ellos, donde:

- Cada **usuario** está representado con una tupla de 2 elementos, donde el primero corresponde al número de identificación (id) y el segundo a su nombre de usuario. Todos los usuarios de la red se encuentran en una lista de `usuarios`.
- Una **publicación** es una tupla de 3 elementos compuesta por: el autor de dicha publicación, el texto publicado y el conjunto de los usuarios que le dieron *me gusta*. Todas las publicaciones de la red están en la lista `publicaciones`.
- La **amistad** entre dos miembros de la red está representada con una tupla de dos usuarios. Las relaciones de amistad se encuentran en la lista `relaciones`.

Luego, podemos definir nuestra **red social** como una tupla de 3 elementos que contenga los tres conceptos antes mencionados, es decir: `(usuarios,relaciones,publicaciones)`

## Definiciones y funciones

Para representar y manipular la red social se cuenta con el archivo `alg1-tp.hs` que tiene las siguientes definiciones de tipos:

- `type Set a = [a]`

- `type Usuario = (Integer, String`[1]`)`
- `type Relacion = (Usuario, Usuario)`
- `type Publicacion = (Usuario, String, Set Usuario)`
- `type RedSocial = (Set Usuario, Set Relacion, Set Publicacion)`

Además, en el archivo `alg1-tp.hs` se definen las siguientes funciones básicas:

- `usuarios :: RedSocial -> Set Usuario`
  Devuelve el conjunto de usuarios.
- `relaciones :: RedSocial -> Set Relacion`
  Devuelve el conjunto de relaciones.
- `publicaciones :: RedSocial -> Set Publicacion`
  Devuelve el conjunto de publicaciones.
- `idDeUsuario :: Usuario -> Integer`
  Devuelve el número de identificación de un usuario.
- `nombreDeUsuario :: Usuario -> String`
  Devuelve el nombre de un usuario.
- `usuarioDePublicacion :: Publicacion -> Usuario`
  Devuelve el usuario de una publicación.
- `likesDePublicacion :: Publicacion -> Set Usuario`
  Devuelve el conjunto de usuarios que le dieron *me gusta* a una publicación.

# Ejercicios obligatorios

Se pide implementar las siguientes funciones:

- `nombresDeUsuarios :: RedSocial -> Set String`
  Dada una red social retorna un conjunto con los nombres de todos los usuarios.
- `amigosDe :: RedSocial -> Usuario -> Set Usuario`
  Dada una red social y un usuario retorna el conjunto de amigos del mismo.
- `cantidadDeAmigos :: RedSocial -> Usuario -> Int`
  Dada una red social y un usuario retorna la cantidad de amigos de dicho usuario.
- `usuarioConMasAmigos :: RedSocial -> Usuario`
  Dada una red social retorna el usuario con más amigos (de existir más de uno devuelve cualquiera).
- `estaRobertoCarlos :: RedSocial -> Bool`
  Dada una red social retorna verdadero si y sólo si algún usuario tiene más de un millón de amigos.
- `publicacionesDe :: RedSocial -> Usuario -> Set Publicacion`
  Dada una red social y un usuario retorna el conjunto de publicaciones del mismo.
- `publicacionesQueLeGustanA :: RedSocial -> Usuario -> Set Publicacion`
  Dada una red social y un usuario retorna el conjunto de publicaciones a las que el usuario les dió like.
- `lesGustanLasMismasPublicaciones :: RedSocial -> Usuario -> Usuario -> Bool`
  Dada una red social y dos usuarios indica si les gustan las mismas publicaciones.
- `tieneUnSeguidorFiel :: RedSocial -> Usuario -> Bool`
  Dada una red social y un usuario, indica si existe otro usuario que le puso *me gusta* a todas las publicaciones del primer usuario.

[1] Cabe aclarar que el tipo `String` es un sinónimo de `[Char]`, es decir una lista de caracteres.

## Ejercicio optativo

- `existeSecuenciaDeAmigos :: RedSocial -> Usuario -> Usuario -> Bool`
  Dada una red social y dos usuarios, indica si existe una secuencia de usuarios relacionados para llegar del primero al segundo.

## Pautas de Entrega

Para la entrega del trabajo práctico se deben tener en cuenta las siguientes consideraciones:

- El trabajo se debe realizar en grupos de tres alumnos.
- El archivo con el código fuente debe tener nombre `alg1-tp.hs`, y debe entregarse mediante el formulario `http://cor.to/TPalgeb-2019c1`. Además, en el archivo entregado debe indicarse, en un comentario arriba de todo: nombre y LU (o DNI) de cada integrante.
- El código debe poder ser ejecutado en el GHCI instalado en los laboratorios del DC, sin ningún paquete especial.
- No está permitido alterar los tipos de datos presentados en el enunciado, ni utilizar técnicas no vistas en clase para resolver los ejercicios (como por ejemplo, alto orden).
- Pueden definirse todas las funciones auxiliares que se requieran. Cada una debe tener un comentario indicando claramente qué hace.
- No es necesario entregar un informe sobre el trabajo.
- La fecha límite de entrega es el viernes 21/6 a las 23:59.

Los objetivos a evaluar para aprobar este trabajo práctico son:

- **Correctitud**: todos los ejercicios obligatorios deben estar bien resueltos.
- **Declaratividad**: el código debe estar comentado y los nombres de las funciones que se definan deben ser apropiados.
- **Consistencia**: el código debe atenerse al uso correcto de las técnicas vistas en clase como recursión o *pattern matching*.
- **Prolijidad**: evitar repetir código innecesariamente y usar adecuadamente las funciones previamente definidas (por el enunciado o por ustedes mismos).
- **Testeo**: es obligatorio entregar además de la resolución de los ejercicios casos de test para cada uno de ellos.

**Importante:** se admitirá un único envío, sin excepción, por grupo. Planifiquen el trabajo para llegar a tiempo con la entrega.

## Referencias del lenguaje Haskell

- **The Haskell 2010 Language Report:** la última versión oficial del lenguaje Haskell a la fecha, disponible online en `http://www.haskell.org/onlinereport/haskell2010`.
- **Learn You a Haskell for Great Good!**: libro accesible, para todas las edades, cubriendo todos los aspectos del lenguaje, notoriamente ilustrado, disponible online en `http://learnyouahaskell.com` y en `http://aprendehaskell.es/` (en español).
- **Real World Haskell**: libro apuntado a zanjar la brecha de aplicación de Haskell, enfocándose principalmente en la utilización de estructuras de datos funcionales en la "vida real", disponible online en `http://book.realworldhaskell.org/read`.
- **Hoogle**: buscador que acepta tanto nombres de funciones y módulos, como signaturas y tipos *parciales*, online en `http://www.haskell.org/hoogle/`