FUNDAMENTOS DE COMPUTACIÓN
ENTREGABLE 1
ENTEROS

Este trabajo tiene un puntaje de 6 puntos, y debe ser realizado SIN ASISTENCIA de herramientas de Inteligencia Artificial. Se debe subir a Aulas antes del día 16/10/25 a las 21:00 hs.

Enteros como pares

Se define el tipo `E` de los números enteros, representados como pares compuestos por un signo y un natural:

```
data Signo where { Pos ::  Signo ; Neg ::  Signo } deriving Show

data Entero where { E ::  Signo -> N -> Entero } deriving Show
```

Así, por ejemplo el número entero -1 se representará con la expresión `E Neg uno`.

Esta no es la mejor definición, ya que el cero va a tener dos representaciones, lo cual deberá ser tenido en cuenta al definir las funciones (se aclarará más adelamte).

Funciones a implementar

En este trabajo entregable se pide definir las siguientes funciones para los números Enteros definidos arriba. Se pueden utilizar las funciones definidas en el Laboratorio 2 para las instancias `Eq, Ord` y `Num` de los Naturales que considere necesarias, las cuales deberán ser importadas de un archivo llamado `Naturales.hs` (leer instrucciones al final):

1) Implemente la instancia de `Eq` para `Signo`, o sea: defina la igualdad de signos.
   Ejemplos:
   ```
   Pos == Pos = True
   Neg == Pos = False
   ```

2) Implemente la instancia de `Eq` para `Entero`, o sea: defina la igualdad entre números enteros, teniendo en cuenta que el cero puede tener diferentes representaciones.
   Ejemplos:
   ```
   E Pos dos == E Pos dos = True
   E Pos dos /= E Pos tres = True
   E Pos dos == E Neg dos = False
   E Pos O == E Neg O = True
   ```

3) Implemente la instancia de `Ord` para `Signo`, o sea: defina la función `(<=)` para los signos, de forma tal que `Neg < Pos`.
   Ejemplos:
   ```
   Neg <= Pos = True
   Pos > Neg = True
   ```

4) Implemente la instancia de `Ord` para `Entero`, o sea: defina la función `(<=)` para los números enteros.
   Ejemplos:
```
E Neg tres <= E Neg dos = True
E Neg dos > E Pos uno = False
E Pos cinco < E Pos (S cinco) = True
E Pos O <= E Neg O = True -- tener en cuenta este caso.
```

5) Implemente la instancia de `Num` para `Entero`, o sea: defina las funciones `(+)`, `(*)` y `(-)` para los números enteros. Cuando el resultado de una operación es cero, puede devolver indistintamente el cero positivo o el negativo.
   Ejemplos:
```
E Pos uno + E Pos dos = E Pos (S(S(S O)))
E Neg uno + E Neg dos = E Neg (S(S(S O)))
E Pos cinco + E Neg tres = E Pos (S(S O))
E Neg dos + E Pos dos = E Pos O o E Neg O (ambos son resultados correctos)
E Pos dos * E Neg dos = E Neg (S(S(S(S O))))
E Neg dos * E Neg uno = E Pos (S(S O))
E Pos dos - E Pos cinco = E Neg (S(S(S O)))
E Pos dos - E Pos dos = E Pos O o E Neg O (ambos son resultados correctos)
```

Entregables

- El trabajo puede realizarse en forma individual o en parejas.
- La entrega deberá realizarse por Aulas antes del **16/10/25 a las 21:00 hs.**
- Deberan subirse a Aulas dos archivos Haskell:
  - `Naturales.hs`, con la definición de los Naturales y las funciones del laboratorio 2 que hayan definido para `N` y que sean necesarias para la solución (como las instancias de Eq, Ord, Num, etc. para los Naturales), y
  - `Enteros.hs`, con el código fuente de la solución.
- Deberá hacerse **una única entrega por equipo**. Los archivos deben incluir los nombres y números de estudiantes como comentarios al principio de los mismos.
- En Aulas se encuentra el archivo `Enteros.hs` con las funciones que deben implementarse. Para facilitar la corrección deberá usarse este último como template.
- No se corregirán archivos que no compilen, por lo que recomendamos comentar el código que no compile y dejar como `undefined` las funciones no implementadas.
- Se utilizarán herramientas para la detección de copias y de uso de herramientas de IA, por lo que recomendamos NO COMPARTIR CÓDIGO, ya que las copias serán penalizadas.
- Para comprobar autoría, este entregable tendrá una instancia de defensa durante el 1º parcial.