# Ejercicio 1

```haskell
{-# LANGUAGE GADTs, EmptyDataDecls, EmptyCase #-}
{-# OPTIONS_GHC -fno-warn-tabs #-}

module Entregable2 where
  import Prelude (Show)
  data Bool where {False :: Bool; True :: Bool } deriving Show

  -- funcion not disponible
  not :: Bool -> Bool
  not = \b -> case b of {
                    False -> True;
                    True -> False
  }

  -- funcion distintos devuelve True cuando los 3 Bool no son iguales
entre si
  distintos :: Bool -> Bool -> Bool -> Bool
  distintos = \b1 -> \b2 -> \b3 -> case b1 of {
                                      False -> case b2 of {
                                                   False -> b3;
                                                   True -> True
                                      };
                                      True -> case b2 of {
                                                   False -> True;
                                                   True -> not b3
                                      };
  }
```

# Ejercicio 2

Demostrar que `distintos b b b = False` para todo `(b :: Bool)`.

Tabla de verdad (resumida) para la función `distintos`:

| b1 | b2 | b3 | distintos b1 b2 b3 |
|---|---|---|---|
| True | True | True | False |
| False | False | False | False |
| Otros | - | - | True |

## Demostración por casos en `b :: Bool`

### Caso `b = False` : `distintos False False False =? False`

```
distintos False False False =? False
    Aplico def. distintos, beta x3, case x2

False = False
    Por reflexividad del =
```

### Caso `b = True` : `distintos True True True =? False`

```
distintos True True True =? False
    Aplico def. distintos, beta x3, case x2

False = False
    Por reflexividad del =
```

# Ejercicio 3

Demostrar que `distintos b1 b1 b2 = distintos b2 b2 b1` para todo `(b1 :: Bool)` y `(b2 :: Bool)`

## Demostración por casos en `(b1 :: Bool)` para `(b2 :: Bool)` cualquiera

### Caso `b1 = False` para todo `(b2::Bool)`: `distintos False False b2 =? distintos b2 b2 False`

#### Demostración por casos en `(b2::Bool)`

- Caso `b2 = False`: `distintos False False False =? distintos False False False`

```
distintos False False False =? distintos False False False
    Por reflexividad del = , son expresiones identicas

distintos False False False = distintos False False False
```

- Caso `b2 = True`: `distintos False False True =? distintos True True False`

```
distintos False False True =? distintos True True False
    Aplico def. distintos, beta x3, case x2 de ambos lados

True = True
    Por reflexividad del = , son expresiones identicas
```

### Caso `b1 = True` para todo `(b2::Bool)`: `distintos True True b2 =? distintos b2 b2 True`

#### Demostración por casos en `(b2::Bool)`

- Caso `b2 = False`: `distintos True True False =? distintos False False True`

```
distintos True True False =? distintos False False True
    Aplico def. distintos, beta x3, case x2 de ambos lados

True = True
    Por reflexividad del = , son expresiones identicas
```

- Caso `b2 = True`: `distintos True True True =? distintos True True True`

```
distintos True True True =? distintos True True True
    Por reflexividad del = , son expresiones identicas

distintos True True True = distintos True True True
```