Nombre: Antonio J. Galán Herrera

1. Determinar el tipo de las siguientes funciones

```
subst f g x = f x (g x)
cross (f,g) (x,y) = (f x, g y)
```

```
subst :: (a -> b -> c) -> (a -> b) -> a -> d
cross :: ((a -> c),(b -> d)) -> (a,b) -> (c,d)
```

2. Da un habitante para el tipo

```
(a -> b -> c) -> b -> a -> c
```

```
función g y x = g (h x) + y
```

3. Escribe el tipo de la función

```
f x y z t
      | z < t = x == y
      |otherwise = False
```

```
f :: (Eq a, Eq b, Ord c, Ord d) => a -> b -> c -> d -> Bool
```

4. Escribe en haskell la función de Ackermann definida por:

$$A(m,n) = \begin{cases} n+1 & m == 0 \\ A(m-1,1) & m > 0 \& n == 0 \\ A(m-1,A(m,n-1)) & m > 0 \& n > 0 \end{cases}$$

```
ackermann :: (Integer,Integer) -> Integer
ackermann (m,n) | m == 0           = n+1
                | m > 0 && n == 0  = ackermann(m-1,1)
                | m > 0 && n > 0   = ackermann(m-1,ackermann(m,n-1))
                | m < 0 && n >= 0  = error "El valor m no está en el dominio."
                | m >= 0 && n < 0  = error "El valor n no está en el dominio."
                | m < 0 && n < 0   = error "Valores m y n fuera del dominio."
```

5. Escribe la función

```
cerosUnos :: Integer -> (Integer, Integer)
```

que toma como argumento un número binario y devuelve un par donde la primera componente cuenta el número de ceros que hay en el argumento y la segunda componente cuenta el número de unos.

```
Main> cerosUnos 1010101010000011101
(10,9)
```

```
cerosUnos :: Integer -> (Integer,Integer)
cerosUnos 0 = (1,0)
cerosUnos 1 = (0,1)
cerosUnos x | mod x 10 == 1 = (fst (cerosUnos num), 1 + snd (cerosUnos num))
            | mod x 10 == 0 = (1 + fst (cerosUnos num), snd (cerosUnos num))
            | otherwise     = error "¿Eso es un número binario?"
    where
        num = div x 10
```