Ejercicio 1 ★

Considerar las siguientes definiciones de funciones:

```
- max2 (x, y)  | x >= y = x                    - evaluarEnCero = \f -> f 0
               | otherwise = y                 - dosVeces = \f -> f . f
- normaVectorial (x, y) = sqrt (x^2 + y^2)     - flipAll = map flip
- subtract = flip (-)                          - flipRaro = flip flip
- predecesor = subtract 1
```

I. ¿Cuál es el tipo de cada función? (Suponer que todos los números son de tipo `Float`).

II. Indicar cuáles de las funciones anteriores *no* están currificadas. Para cada una de ellas, definir la función currificada correspondiente. Recordar dar el tipo de la función.

```
max2::(Float, Float)-> Float                                    no currificada
normaVectorial::(Float, Float)-> Float                          no currificada
subtract::  (Num t1) => t1 -> t1 -> t1
predecesor::  Num t1 => t1 -> t1
evaluarEnCero::  (Num t1) => (t1 -> t2) -> t2
evaluarDosVeces::  (t1 -> t1) -> t1 -> t1
flipAll::  [a->b->c]->[b->a->c]
flipRaro::  b->(a->b->c)->a->c


max2Curry::(Num a)=> a -> a -> a
max2Curry x y
| x <= y = y
| otherwise = x


normaVectorialCurry::  Floating a => a -> a -> a      ⟵ No se porque es floating pero asi lo dice ghci
normaVectorialCurry x y = sqrt (x*x+y*y)
```

Ejercicio 2 ★

I. Definir la función `curry`, que dada una función de dos argumentos, devuelve su equivalente currificada.

II. Definir la función `uncurry`, que dada una función currificada de dos argumentos, devuelve su versión no currificada equivalente. Es la inversa de la anterior.

III. ¿Se podría definir una función `curryN`, que tome una función de un número arbitrario de argumentos y devuelva su versión currificada?
Sugerencia: pensar cuál sería el tipo de la función.

! ! ! Solucion asumiento que por una funcion de dos argumentos nos referimos a una dupla

```
curry::((a,b)->c) -> a -> b -> c
curry f x y = f (x,y)

uncurry::  (a -> b -> c) -> (a,b) -> c
uncurry f (x,y) = f x y
3- No...?  (No me queda claro, pensando en el tipo como se representaria una arbitraria
cantidad de argumentos?)
Ejercicio 3 en un .hs
```