Práctica 3: Datatypes

Se utilizará la siguiente definición de los números naturales
`datatype nats = Zero | Succ of nats`

1. Definir la función `sumaNat`, que toma dos valores de tipo `nats` y devuelve un `nats` con la suma de ambos. Aplicar pattern matching sobre el segundo argumento.
2. Definir la función `duplicarNat` que toma un valor de tipo `nats` y devuelve un `nats` que representa el doble del argumento.
3. Definir la función `prodNat`, que toma dos valores de tipo `nats` y devuelve un `nats` con el producto de ambos. Aplicar pattern matching sobre el segundo argumento.
4. Definir la función `powerNat`, que toma dos valores de tipo `nats` y devuelve un `nats` con valor de elevar el primero a la potencia del segundo. Aplicar pattern matching sobre el segundo argumento.
5. Definir la función `factNat`, que toma un valor de tipo `nats` y devuelve el factorial representado en `nats`.
6. Definir la función `fibNat`, que toma un valor de tipo `nats` y devuelve el valor correspondiente de la secuencia de Fibonacci para ese valor representado en `nats`. Secuencia de Fibonacci: fib(0) = 0 fib(1) = 1 fib(n+2) = fib(n) + fib(n+1)
7. Defina funciones de conversión entre `nats` e `int`, es decir: `nats2int`: función que toma un `nats` y devuelve un `int` `int2nats`: función que toma un `int` y devuelve un `nats`
8. Definir la función `leq` (Less or EQual, menor igual), que toma dos `nats` y devuelve un booleano que defina si el primer `nats` es menor o igual al segundo `nats`.
9. Definir las funciones `lt` (lower than, menor), `gt` (greater than, mayor) y `geq` (greater or equal, mayor igual), similares a la anterior. No utilice pattern matching, parta de la función definida en el apartado 7.