Programación para la Inteligencia Artificial

Haskell - Práctica 1
Funciones y reducción de expresiones

Usa los ejemplos de programas Haskell de los apuntes del tema 4, en el fichero `Tpia4funciones.hs`, y el fichero `fac.hs` (que contiene múltiples versiones de la función *factorial*) para probar el entorno GHCi.

Puedes también ayudarte del documento "*Glasgow Haskell Compiler.* User's Guide.", que puedes encontrar en:

`https://downloads.haskell.org/ghc/latest/docs/html/users_guide/`

Resuelve, además, los siguientes ejercicios:

1. Definir una función `maximo` que devuelva el mayor de sus dos argumentos. Definir la función de forma no currificada y currificada.

2. Definir una función para calcular el área de un círculo, dado su radio r (usar 22/7 como aproximación de π).

3. Los números de Fibonacci $f_0, f_1, ...$ se definen de la siguiente forma: $f_0 = 0$, $f_1 = 1$ y $f_{n+2} = f_n + f_{n+1}$ para todo $n \geq 0$. Dar la definición de la función `fib` que toma un entero n y devuelve f_n. Hacerlo de dos formas diferentes (una de ellas, al menos, por ajuste de patrones).

4. Definir una función `absol :: Integer -> Integer` que devuelva el valor absoluto de un entero.

5. Definir las funciones `aumentar` y `aumentar2` de tipo:

   $$aumentar, aumentar2 :: Integer -> Integer$$

   de manera que `aumentar` devuelva el cuadrado de $(x + 1)$ y `aumentar2` el siguiente del cuadrado de x.

   Definir la función `sucesor` y volver a definir `aumentar` y `aumentar2` como composición de las funciones `cuadrado` y `sucesor` sin usar la variable de entrada.

6. Dar dos definiciones diferentes para la función `nAnd`:

   ```
   nAnd ::  Bool -> Bool -> Bool
   ```

7. Dar, al menos, dos definiciones de la función `xOr` definida mediante la tabla siguiente:

a	b	a xor b
False	False	False
False	True	True
True	False	True
True	True	False

8. Dar la definición de una función `minimoTres` que calcule el mínimo de tres números.

9. Usar las guardas para definir la función de tres variables `maximoTres` que calcule el máximo de tres números.

10. Definir la función `numeroCentral` tal que, dados tres números, devuelva aquel cuyo valor es intermedio al de los otros dos.

11. Definir la función `productoRango` que devuelve, dados dos números naturales m y n, el producto:

    $$m * (m + 1) * ... * (n - 1) * n$$

    En caso de que $n < m$ debe devolver 0.

    Definir la función `factorial` usando `productoRango`.

12. Usando la función suma sobre los naturales, dar una definición recursiva de la multiplicación de números naturales.