Programación Declarativa. Grado Matemáticas. UCM.
HOJA 1 DE EJERCICIOS

1. Escribe una función que dado el radio de un círculo devuelva su área. (Tanto en esta como en todas las demás funciones que se pidan en esta hoja, indica también cuál es el tipo de la función definida.)

2. Escribe una función que dado un entero devuelva un booleano indicando si el entero pasado como parámetro es un cuadrado perfecto o no. (Puedes usar la función `round` como paso intermedio.)

3. Escribe una función que calcule el número de raíces reales distintas de una ecuación de segundo grado. Los datos de entrada a la función serán los coeficientes de la ecuación.

4. Construye una expresión que tenga dos apariciones de la lista vacía, siendo la primera de ellas de tipo [Int] y la segunda de tipo [Char].

5. Para cada una de las siguientes expresiones, indica si son válidas o no desde el punto de vista de los tipos. En caso de que sean válidas, indica cuál es su tipo:

```
[7,  9]    7:9      [(7,9)]     [[7],[9]]     ([7], [9])
7: [9]     [7]:9    (7, [9])    [7, [9]]      [7:[9]]
```

6. Razona cuáles de las siguientes igualdades son válidas, explicando qué tipo debe suponerse en cada caso para la variable `xs` de modo que las expresiones tengan un tipo correcto:

```
[[]] ++ xs = xs           [[]] ++ xs = [xs]     [[]] ++ xs = [[], xs]
[[]] ++ [xs] = [[], xs]   [xs] ++ [] = [xs]     [xs] ++ [xs] = [xs, xs]
```

7. Escribe una función `zip3` que se comporte como la función `zip` (es decir, la función `cremallera` vista en clase) pero juntando tres listas de entrada para que generen una única lista de tuplas de salida.

8. Escribe una función que calcule el productorio de todos los elementos de una lista de enteros. Es decir, dada la lista [3,5,1,5,2] debe devolver 3*5*1*5*2, que es 150.

9. Escribe una función que calcule el producto escalar de dos listas. Es decir, dadas las listas [3,5,2] y [1,5,8] debe devolver 3*1+5*5+2*8, que es 44.

10. Escribe una función recursiva que se comporte igual que `++`.

11. Escribe una función `concatena` que reciba una lista de listas y devuelva una única lista que contenga el resultado de concatenar todas las listas de entrada. Por ejemplo, para [[1,3,4],[2,8],[],[3,6]] debe devolver [1,3,4,2,8,3,6].

12. Escribe una función que dada una lista de entrada, devuelva un booleano que indique si la lista de entrada estaba o no ordenada de menor a mayor.

13. Escribe una función que dada una lista de enteros de entrada devuelva esa misma lista pero eliminando de ella todos los números pares. Es decir, para la lista [1,34,12,7,8,1,9] devolverá [1,7,1,9]. Escribe otra función que haga lo mismo pero dejando en la lista sólo los números pares.

14. Suponiendo que la lista de entrada tiene igual cantidad de números pares e impares, escribe una función `coloca` tal que `coloca xs` devuelva una permutación de la lista `xs` en la que aparezcan alternativamente números pares e impares.

15. Escribe una función que dado un entero devuelva el número de bits que serían necesarios para representar dicho número entero, suponiendo que se usa un bit para el signo. (Por ejemplo, para 5 debe devolver 4, pues se necesita un bit para el signo y otros tres para representar 5 en binario: 101).

16. Escribe una función que dado un número entero positivo devuelva una lista de enteros que contenga la representación binaria (sin signo) del entero de entrada. Por ejemplo, para 13 debe devolver [1,1,0,1].

17. Escribe una función que ordene una lista utilizando el método de ordenación por selección. Escribe otra función que ordene una lista utilizando el método de ordenación rápida (es decir, el quicksort ideado por C.A.R. Hoare).

18. Define una función `takeLast :: Int ->[a] ->[a]` que se comporte de forma similar a la función `take` (es decir, la función `coger` vista en clase) pero cogiendo los elementos de derecha a izquierda. Por ejemplo, `takeLast 3 [1,3,5,2,8,7]` debe devolver `[7,8,2]`.

19. Repite el ejercicio anterior pero para hacer una función `dropLast`.

20. Dada la función

    ```
    genFib :: Int -> Int -> [Int]
    genFib m n = m : genFib n (m+n)
    ```

    indica cómo puedes utilizarla para definir una función no recursiva que se comporte igual que la función `fibonacci` vista en clase.

21. Escribe una función que calcule el máximo común divisor de dos enteros positivos. Escribe otra que calcule el mínimo común múltiplo.

22. Suponiendo que para representar números racionales $\frac{a}{b}$ usamos tuplas de dos enteros (a, b), escribe funciones para hacer la suma, la resta, el producto y la división de números racionales. Todas ellas deben devolver el resultado simplificado, de modo que nunca se devuelva $(12, 8)$, sino $(3, 2)$. (Define una función auxiliar que simplifique un número racional, de modo que puedas utilizar dicha función en las otras cuatro.)

23. Usando la misma representación de números racionales que en el ejercicio anterior, escribe una función que eleve un número racional a un número entero cualquiera. (Fíjate en que se deben permitir potencias negativas.)

24. Escribe una función que devuelva el máximo elemento de una lista no vacía. Haz otra función análoga pero que devuelva el mínimo.

25. Escribe una función que devuelva el máximo común divisor de todos los números incluidos en una lista no vacía. Haz otra análoga pero que devuelva el mínimo común múltiplo.