Práctica 9: Entorno local - Uso de `let`

Para resolver los siguientes ejercicios debe utilizar expresiones `let`.

1. Escriba una función que tome un valor entero `x` y calcule $\texttt{x}^{1000}$.

2. Dada la siguiente función:

```
- fun split(nil)= (nil, nil)
      | split([a]) = ([a], nil)
      | split(a::b::cs) =
           let
                val (M,N) = split(cs)
           in
                (a::M, b::N)
           end;
val split = fn : 'a list ->'a list * 'a list
```

   reescriba la función `split` para que no use pattern matching en la declaración de variable, es decir, reemplace la línea
   `val (M,N) = split(cs)`
   por
   `val x = split(cs)`
   y obtenga las componentes de `x` cuando sea necesario.

3. Escriba una función que permita obtener el valor máximo de una lista de reales (*Ayuda: calcule primero el máximo de la cola de la lista*).

4. Escriba una función que tome una lista de pares de enteros y sume cada elemento de los pares por separado, es decir, debe devolver un par cuya primera componente sea la suma de todas las primeras componentes de los pares de la lista y la segunda componente sea la suma de todas las segundas componentes de dichos pares.

5. Escriba una función que calcule $\texttt{x}^{2^i}$ para un valor real de `x` y un entero no negativo i. Debe haber una única llamada recursiva en su definición.

6. Escriba una función que tomando una lista de enteros devuelva una tupla de dos valores. El primer elemento de la tupla debe coincidir con la suma de los elementos en las posiciones pares de la lista y el segundo elemento debe coincidir con la suma de los elementos en las posiciones impares. No use funciones auxiliares.

7. Escriba una versión de la función `filter` que tome un solo argumento, el predicado `P`, y que genere como resultado una función que tomando una lista de elementos de un tipo determinado, devuelva otra lista compuesta únicamente por aquellos elementos de de la lista que satisfacen el predicado `P`.