Paradigmas de Programación

Práctica 4

Nota Importante:

Realice las implementaciones del ejercicio 1 en un fichero de nombre `power.ml` y las del ejercicio 2 en un fichero de nombre `mcd.ml`. Allí donde se pidan explicaciones y razonamientos, inclúyalos en estos mismos ficheros entre comentarios.

Cuando se solicite la entrega de esta práctica, cada alumno deberá subir a su repositorio de prácticas (del cual se indicará su ubicación más adelante) un directorio `p4` cuyo contenido debe ser únicamente los ficheros `power.ml` y `mcd.ml`.

Sea muy cuidadoso a la hora de crear el directorio y los ficheros, y **respete los nombres indicados**. En particular, fíjese que todos estos nombres sólo contienen letras en minúsculas, números y puntos.

Además, **todos los ficheros deben compilar sin errores** con las siguientes órdenes:

```
ocamlc -c power.ml
ocamlc -c mcd.ml
```

Ejercicios:

1. Si x es un número entero cualquiera e y es número entero mayor que 0, se cumple la siguiente propiedad:

   $$x^y = x \times x^{y-1}$$

   Utilizando directamente esta propiedad, defina en OCaml (recursivamente) una función `power: int -> int -> int` tal que, para cualesquiera `x:int`, `y:int`, `y >= 0`, `power x y` tenga el valor de x^y. Por ejemplo, `power 2 10 = 1024`.

   También son ciertas las siguientes propiedades (razónelas desde el punto de vista matemático):

   $$x^y = (x \times x)^{y/2}, \quad \text{si } y \text{ es par}$$
   $$x^y = x \times (x \times x)^{y/2}, \quad \text{si } y \text{ es impar}$$

   Utilice directamente estas dos propiedades para definir en OCaml una función `power': int -> int -> int`, que sea una versión mejorada (en términos de eficiencia) de la definición anterior.

   Explique por qué `power'` debería ser mejor que `power` en términos de eficiencia y razone si realmente merece la pena la ganancia obtenida al estar operando en `int` (y no en $\mathbb{Z}$).

   Todo lo anterior sería igualmente válido para potencias de base real y exponente natural. Defina una función `powerf: float -> int -> float`, tal que `powerf x n` tenga el valor de x^n, para cualesquiera `x:float`, `n:int`, `n >= 0`.

2. Implemente en OCaml una función `mcd: int * int -> int`, tal que, para cualesquiera `x:int`, `y:int`, `x >= 0`, `y >= 0`, `(x <> 0 || y <> 0)`, `mcd (x,y)` sea el máximo común divisor de `x` e `y`.

   Defina (recursivamente) esta función basándose en las siguientes propiedades:

   $$mcd(x, y) = mcd(y, x)$$
   $$mcd(x, y) = mcd(x \ mod \ y, y), \text{si } y > 0$$