# PROGRAMACIÓN DECLARATIVA

22 de diciembre 2006

NOMBRE: _____

I.I. ☐ I.T.I.G ☐

**1.** *(4 puntos)* Escriba el resultado de la compilación y ejecución de las siguientes frases, con tipos y valores, como lo indicaría el *toplevel* de *ocaml*:

```
let x, y = "hola", "adiós";;
```

```
let x y = x ^ y;;
```

```
let x = x "pepe" in x;;
```

```
let x::y::z = [1] @ [2] @ [3];;
```

```
let x::y::z = [1]::[2]::[3]::[];;
```

```
let rec m2 l1 l2= match (l1,l2) with
    ([],[]) -> [] | (a::b, c::d) -> a c :: m2 b d;;
```

```
m2 [abs] [1; -2];;
```

```
(function x -> x (x 2)) (function x -> 2 * x * x);;
```

**2.** Diremos que una lista es *sub-lista* de otra si puede obtenerse eliminando algunos elementos de esta. Puesto que cada elemento de una lista de n elementos puede ser o no eliminado para obtener una sub-lista, esta tendrá $2^n$ sub-listas. Así, por ejemplo, la lista [3;5;3] tendría las siguientes sub-listas: [], [3], [5], [5; 3], [3], [3; 3], [3; 5] y [3; 5; 3]. (Nótese que alguna sub-lista aparece más de una vez debido a que puede ser obtenida eliminando distintos elementos de la lista original).

**a.** *(2 puntos)* Defina una función *sublistas : 'a list -> 'a list list* que dé todas las sublistas de una lista dada. (Si lo desea puede optar por no incluir repeticiones en la lista de sublistas; pero en ese caso debe indicarlo explícitamente).

**b.** *(2 puntos)* Defina una función *sublista_de : 'a list -> 'a list -> bool*, tal que *sublista_de l1 l2* indique si *l2* es, o no, *sub-lista* de *l1*. (Se tendrá en cuenta la eficiencia de la definición).

**3.** *(2 puntos)* Defina, utilizando sólo **recursividad terminal**, una función *apariciones: 'a -> 'a list ->int* que devuelva el número de veces que aparece un valor en una lista.