Exámenes 2013, preguntas y respuestas

Paradigmas de Programación (Universidade da Coruña)

JULIO DE 2013

1) escriba el resultado de la compilación tal y como lo haría el Ocaml:

```
let rec p = function 0 -> true |n -> i(n-1)
and i = function 0 -> false |n -> p(n-1);;
```

```
val p : int -> bool = <fun>

val i : int -> bool = <fun>
```

```
p 2, i (-2);;
```

```
recursividad infinita
```

```
let p n = p(abs n) and i n = i(abs n) in p 2, i (-2);;
```

```
-: bool * bool = (true,false)
```

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

```
val x: int = 1

val y: int = 2

val z: int list = []
```

```
z::[];;
```

```
(* -: int list list = [[]] *)
```

```
let x,y = y,x;;
```

```
(* val x: int = 2

   val y: int = 1 *)
```

```
let f z = z + 2 * y;;
```

```
(* val f: int -> int = <fun> *)
```

```
f x + f y;;
```

```
(* -: int = 7 *)
```

```
let y = x + y;;
```

```
(* val y: int = 3 *)
```

```
f x + f y;;
```

```
(* -: int = 9 *)
```

```
let p = let x,y = x+y,x-y in y,x;;
```

```
(* val p: int*int = (-1,5) *)
```

```
let p = x+y,y-x in let x,y = p in y,x;;
```

```
(* -: int * int = (1,5) *)
```

2) Pasar la funcion anterior a una sin referencias, de programación imperativa a declarativa.

```
let f (x,y)=
  let x = abs x and y = abs y in
  let
      a = ref (max x y) and
      b = ref (min x y) in
  while !b <> 0 do
      let temp = !a mod !b in a:= !b;
      b:= temp
  done;
  !a;;
```

```
    let f (x,y) =
```

```
        let rec aux = function

            (a,0) -> a

            |(a,b) -> aux(b, a mod b)

        in aux(max (abs x) (abs y),min (abs x) (abs y));;
```

3) Dada la siguiente definicion, hacer el recorrido en anchura del arbol
'a arb -> 'a list.

```
type 'a arb = R of 'a
              |U of 'a * 'a arb
              |B of 'a * 'a arb * 'a arb;;
```

```
    let anchura arbol =

        let rec aux = function

            [] -> []

            |R(x)::t -> x:: aux t

            |U(x,i)::t -> x:: aux (t @ [i])

            |B(x,i,d)::t -> x:: aux (t@[i;d])

        in aux [arbol];;
```

PARADIGMAS DE LA PROGRAMACIÓN
16 de diciembre de 2013

NOMBRE:_____DNI:_____

1. (3 ptos.) Las siguientes frases corresponden al código introducido en una sesión de trabajo con el toplevel de ocaml. Indique, después de cada una de ellas, cuál sería la respuesta que daría el compilador.

```
# let f x = x + 1, x - 1;;
```

```
val f: int → (int * int) = <fun>
```

```
# let x = f 0;;
```

```
val x: (int * int) = (1,-1)
```

```
# let y, x = x and z = x;;
```

```
val y: int = 1
val x: int = -1
val z: (int * int) = (1,-1)
```

```
# let x y = y::[] in x y;;
```

```
-: int list = [1]
```

```
# let mas f g = function (x,y) -> f x + g y;;
```

```
val mas: ('a → int) → ('b → int) → ('a * 'b) → int = <fun>
```

```
# let x = let x = [1;2;3] in List.tl x;;
```

```
val x: int list = [2;3]
```

2. (2 ptos.) Escriba una definicion recursiva terminal de la siguiente función **f: int -> int**

```
let rec f x = if x >= 4 then 3 * f(x-1) - 2 * f(x-3)
              else x;;
```

```
let f x =
  let rec aux n r1 r2 r3 =
       if n<=x then aux (n+1) r2 r3 (3*r3 - 2*r1)
               else r3
  in if x>=4 then aux 4 1 2 3
             else x;;
```

3. (2 ptos.) Escriba una definición recursiva terminal de la función **lista2arbol: 'a list -> 'a arbol**:

```
let rec lista2arbol = function
           [] -> arbol_vacio
       |   h::t -> insert h (lista2arbol t);;
```

donde arbol_vacio: 'a arbol y insert: 'a -> 'a arbol -> 'a arbol. (Puede suponerse que la función insert está definida de modo terminal).

```
let lista2arbol l =
  let rec aux r = function
     []    -> r
  | h::[] -> insert h r
  | h::t  -> aux (insert h r) t
  in aux arbol_vacio (List.rev l);;
```