Examen 15 Enero 2015, preguntas

Paradigmas de Programación (Universidade da Coruña)

PARADIGMAS DE PROGRAMACIÓN

15 / 01 / 2015

APELLIDOS _____ NOMBRE _____
APELIDOS NOME

1. *(3 puntos)* Indique el efecto de la compilación y ejecución de las siguientes frases en ocaml (con tipos y valores, y distinguiendo definiciones de expresiones), como lo indicaría el compilador interactivo:

Indique o efecto da compilación e execución das seguintes frases en ocaml (con tipos e valores, e distinguindo definicións de expresións), como o indicaría o compilador interactivo:

```
let x = [(1,2);(3,4)];;
```

```
let x::y = x in x::x::y;;
```

```
let (x,y)::_ = List.tl x;;
```

```
x + (let x = x + y in x + y);;
```

```
(function x -> x,x) "hola";;
```

```
let rec prod x = function
    [] -> []
  | h::t -> (x,h):: prod x t;;
```

```
prod 1 [2;3;4];;
```

```
let rec lprod l1 l2 = match l1 with
    [] -> []
  | h::t -> prod h l2 @ lprod t l2;;
```

```
lprod [1;2] ['a';'b';'c'];;
```

2. *(0,5 puntos)* Redefina la función *prod* del ejercicio anterior con la función *List.map*, sin utilizar directamente recursividad.

Redefina a función *prod* do exercicio anterior coa función *List.map*, sen utilizar directamente recursividade.

```
let prod
```

3. a) *(0,5 puntos)* Indique el tipo de la función *f* definida a continuación:

Indique o tipo da función *f* definida a continuación:

```
let rec f = function x::y::t -> f (y::t) && x <= y
                   | _ -> true;;
```

```
f:
```

b) *(0,5 puntos)* ¿Es esa definición recursiva terminal? Si no lo fuera, redefina la función *f* de modo que sólo se utilice recursividad terminal.

É esa definición recursiva terminal? Se non o fose, redefina a función *f* de modo que só se utilice recursividade terminal.

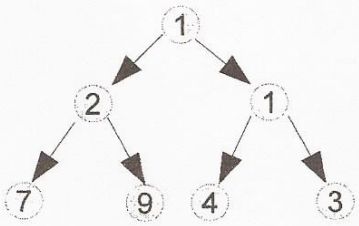
4. Considere la siguiente definición de tipos de datos que sirven para representar árboles estrictamente binarios, con nodos etiquetados con valores de tipo *'a*.

Considere a seguinte definición de tipos de datos que serven para representar árbores estrictamente binarias, con nodos etiquetados con valores de tipo *'a*.

```
type 'a tree = L of 'a | T of 'a * 'a tree * 'a tree
```

a) *(0,5 puntos)* Represente, con un valor de tipo *int tree*, el arbol, *t* , dibujado a continuación.

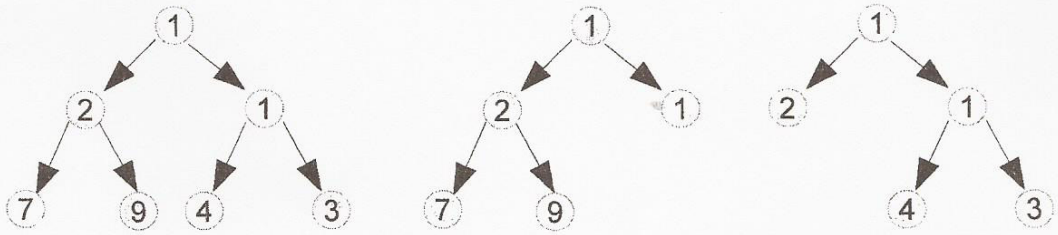
Represente, cun valor de tipo *int tree*, el arbol, *t* , debuxado a continuación.




```
let t =
```

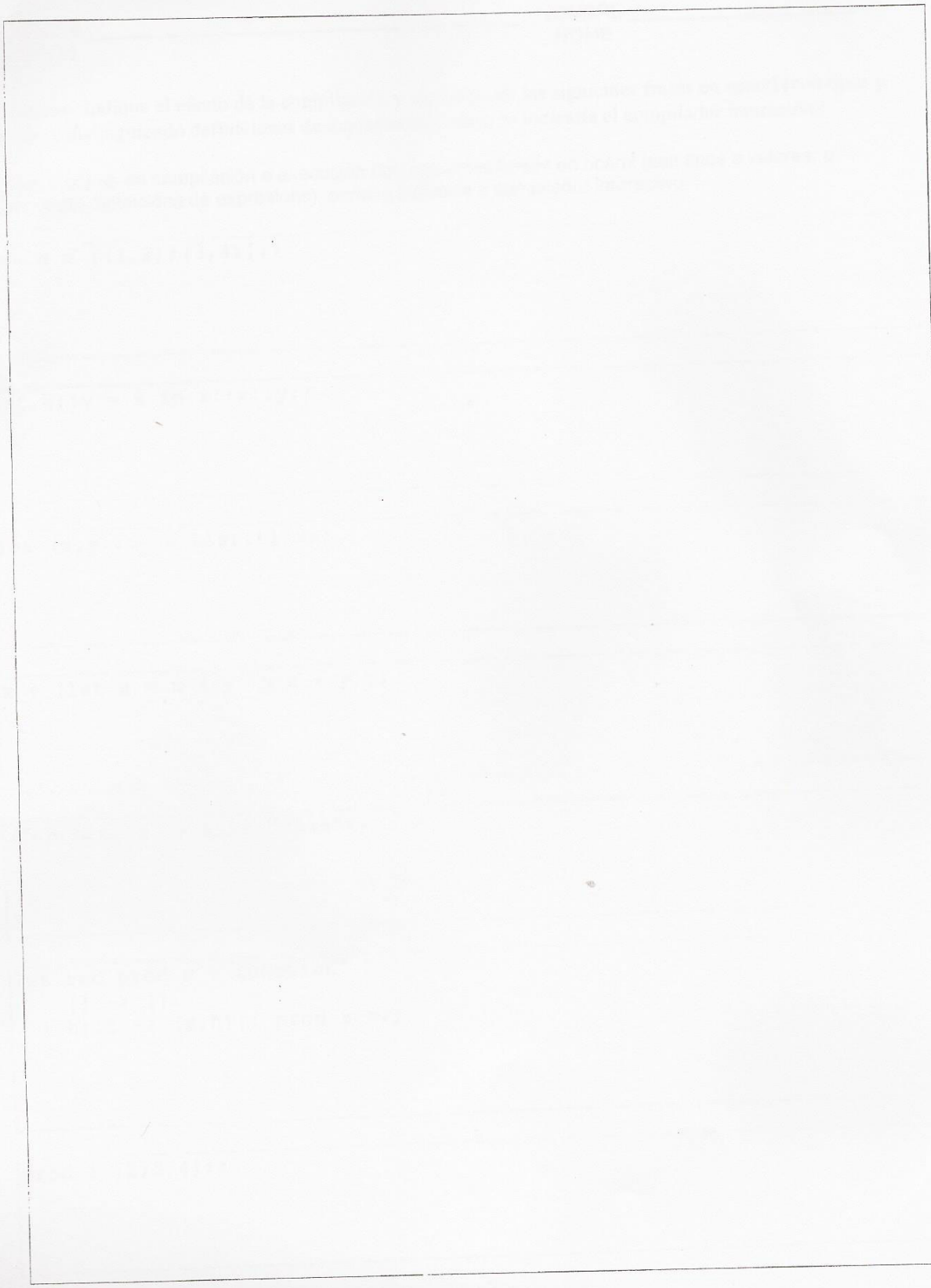
Llamaremos "segmentos raíz" de un árbol, a todos aquellos árboles que puedan obtenerse a partir de este "podándolo", o no, por uno o varios nodos. Siendo más precisos, los segmentos raíz del árbol t, serían exactamente los 5 siguientes:

Chamaremos "segmentos raíz" dunha árbore, a todas aquelas árbores que poidan obterse a partir desta "podándoa", quizáis, por un ou varios nodos. Sendo máis precisos, os segmentos raíz da árbore *t*, serían exactamente os 5 seguintes:



b) *(1 punto)* Defina una función ***es_segrz****: 'a tree -> 'a tree -> bool*, de forma que *es_segrz t1 t2* indique si *t1* es (o no es) un segmento raíz de *t2*.

Defina unha función **es_segrz***: 'a tree -> 'a tree -> bool*, de xeito que *es_segrz t1 t2* indique se *t1* é (ou non é) un segmento raíz de *t2*.



PARADIGMAS DE PROGRAMACIÓN

XANEIRO 2015: Parte de obxetos / *ENERO 2015: Parte de objetos*

APELIDOS/*APELLIDOS*: _____ **NOME/*NOMBRE*:** _____

[0.6 puntos] EXERCICIO / *EJERCICIO 1*:

Indique cales das seguintes definicións son ou non correctas. Se son correctas, indique o resultado que devolvería o compilador interactivo (tipos e valores, clases inclusive); se non o son, indique brevemente por que.

Indique cuáles de las siguientes definiciones son o no correctas. Si son correctas, indique el resultado que devolvería el compilador interactivo (tipos y valores, clases inclusive); si no lo son, indique brevemente por qué.

`class claseB pc1 = object`
`  val atributo1 = pc1`
`  method metodo1 ()  = String.length(atributo1)`
`  method metodo2 pm1 = atributo1^pm1`
`end;;`	

`class subclaseB1 pc1 pc2 = object`
`  inherit claseB pc1`
`  val atributo2 = pc2`
`  method metodo2 pm1 = atributo1^atributo2^pm1`
`end;;`	

`class subclaseB2 pc1 pc2 = object`
`  inherit claseB pc1`
`  val atributo2 = pc2`
`  method metodo2 pm1 pm2 =`
`  atributo1^atributo2^pm1^pm2`
`end;;`	

[0.6 puntos] EXERCICIO / *EJERCICIO 2*:

```
class claseJ = object
  val atributo1 = [10; 20; 30]
  method metodo1 ()  = atributo1
  method metodo2 ()  = List.hd atributo1
  method metodo3 ()  = List.tl atributo1
end;;

class subclaseJ2 = object
  inherit claseJ
  val atributo1 = [0]
end;;

class subclaseJ5 = object
  inherit claseJ as super
  method metodo4 () = super#metodo1()
end;;
```

```
class claseM n =
object
  val atributo1 = n::[]
  method metodo3 ()  = atributo1
  method metodo2 ()  = 0
  method metodo1 ()  = [n; (n+1)]
end;;


let j = new claseJ;;
let sj2 = new subclaseJ2;;
let sj5 = new subclaseJ5;;
let m = new claseM 15;;
```

Dadas as definicións anteriores, indique cales das seguintes listas son ou non válidas. No caso de selo, indique o seu tipo resultante; no caso de non selo, indique brevemente por que.

Dadas las definiciones anteriores, indique cuáles de las siguientes listas son o no válidas. En el caso de serlo, indique su tipo resultante, en el caso de no serlo, indique brevemente por qué.

```
[j; m];;
```