

FINAL-PP-2022.pdf



Taurux



Paradigmas de Programación

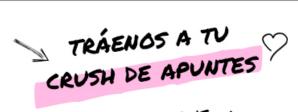


2º Grado en Ingeniería Informática



Facultad de Informática Universidad de A Coruña





LOS QUEME



WUOLAH

ne. F pero no se lo a nadie. val nada, c

NO QUEMES TUS APUNTES

SANA O, 25 € por subir tus apuntes en PDF a Wuolah

te tredas



PARADIGMAS DE PROGRAMACIÓN

27.01.2021

APELLIDOS APELIDOS

NOMBRE NOME

1. (3.5 puntos) Indique el efecto de la compilación y ejecución de las siguientes frases, como lo indicaría el compilador interactivo de OCaml. Distinga claramente expresiones de definiciones.

Indique o efecto da compilación e execución das seguintes frases, como o indicaría o compilador interactivo de OCaml. Distinga claramente expresións de definicións.

let f3 f x = (x, f x, f (f x));; val f3: (a -> a) -> a -> a * a * a = < fun>

let x, y, z = let g x = x * x in f3 g 2;;

val x: int = 2

val y: int = 4

val z: int = 16

(function _ :: _ :: t -> t) [1; 2; 3];;
-: int list = [3]

List.map (function x -> 2 * x + 1);;

-: int list -> int list = <fun>

val f : int list -> int = <fun>

f [1000; 100; 10], f [1000; 100; 10; 1];; -: int * int = (1090, 1089)

List.fold_right (-) [4; 3; 2] 1;;

-: int = 2

* válido
hasta el 3 de
junio de
2022 o hasta
llegar al
tope de
documentos
para esta

promoción

WUOLAH-

 a) (1'5 puntos) Indique el efecto de la compilación y ejecución de las siguientes frases, como lo indicaría el compilador interactivo de OCaml. Distinga claramente expresiones de definiciones.

Indique o efecto da compilación e execución das seguintes frases, como o indicaría o compilador interactivo de OCaml. Distinga claramente expresións de definicións.

```
comb (+);;
-: int list -> int list = <fun>
```

```
comb (+) [1; 2; 3; 4; 5];;
-: int list = [3; 7; 5]
```

b) (1'5 puntos) La implementacion de la función comb definida en el apartado anterior no es recursiva terminal. Redefina la función comb, de modo que su implementación sí sea recursiva terminal.

A implementación da función **comb** definida no apartado anterior non é recursiva terminal. Redefina a función **comb**, de xeito que a súa implementación **sexa recursiva terminal**.

```
let comb f I =
let rec loop f IAux = function
[] -> List.rev IAux
|h::[] -> List.rev (h::IAux)
|h1::h2::t -> loop f (f h1 h2::IAux) t
in loop f [] I;;
```

3. a) (2 puntos) Partiendo de la definición type 'a tree = T of 'a * 'a tree list, indique el efecto de la compilación y ejecución de las siguientes frases, como lo indicaría el compilador interactivo de OCaml. Distinga claramente expresiones de definiciones.

Partindo da definición **type 'a tree = T of 'a * 'a tree list**, indique o efecto da compilación e execución das seguintes frases, como o indicaría o compilador interactivo de OCaml. Distinga claramente expresións de definicións.

```
let s x = T (x,[]);;
- val s: 'a -> 'a tree = <fun>
```

```
let t = T (1,[s 2; s 3; s 4]);;
- val t: int tree = T (1, [T (2,[]); T(3, []); T(4, [])])
```

```
sum t;;
-: int = 10
```

b) (1'5 puntos) La implementacion de la función sum definida en el apartado anterior no es recursiva terminal. Redefina la función sum, de modo que su implementación sí sea recursiva terminal.

A implementación da función **sum** definida no apartado anterior non é recursiva terminal. Redefina a función **sum**, de xeito que a súa implementación **sexa recursiva terminal**.

```
let sum t =
let rec loop aux = function
  T(x,[]) -> aux + x
|T(x,T(x1,I)::t) -> loop (aux+x) (T(x1,List.rev_append I t))
in loop 0 t;;
```