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));;
```

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

```
(function _ :: _ :: t -> t) [1; 2; 3];;
```

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

```
let rec f = function [] -> 0 | h::[] -> h
  | h1::h2::t -> h1 + h2 - f t;;
```

```
f [1000; 100; 10], f [1000; 100; 10; 1];;
```

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

2. **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.

```
let rec comb f = function
    h1::h2::t -> f h1 h2 :: comb f t
  | l -> l;;
```

```
comb (+);;
```

```
comb (+) [1; 2; 3; 4; 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**.

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,[]);;
```

```
let t = T (1,[s 2; s 3; s 4]);;
```

```
let rec sum = function
    T (x, []) -> x
  | T (r, T(r1, l)::t) -> r + sum (T (r1, l @ t));;
```

```
sum t;;
```

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.**