

Enero-2023-resuelto.pdf



Yisha



Paradigmas de Programación



2º Grado en Ingeniería Informática



Facultad de Informática
Universidad de A Coruña



**Que no te escriban poemas de amor
cuando terminen la carrera** ►►►►►►►

☺
*(a nosotros por
suerte nos pasa)*

WUOLAH

Que no te escriban poemas de amor
cuando terminen la carrera ➤➤➤➤➤



WUOLAH

(a nosotros por suerte nos pasa)

No si antes decirte
Lo mucho que te voy a recordar

Pero me voy a graduar.
Mañana mi diploma y título he de
pagar

Llegó mi momento de despedirme
Tras años en los que has estado mi
lado.

Siempre me has ayudado
Cuando por exámenes me he
agobiado

Oh Wuolah wuolah
Tu que eres tan bonita

$$2^8 + 0^7 + 1^4 + 1^5 + 1 = \boxed{7^4}$$

266

PARADIGMAS DE PROGRAMACIÓN

11.01.2023

APELLIDOS
APELIDOS

NOMBRE
NOME

1. (3 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 execução das seguintes frases, como o indicaría o compilador interactivo de OCaml. Distinga claramente expresións de definicións.

let x = let x = 3 in x * x;;
val x : int = 9

1

x + let x = x + 1 in x * x;;
-: int = 109

1

function x -> x;;
-: 'a -> 'a = <fun>

1

let app x f = f x;;
val app : 'a -> ('a -> 'b) -> 'b = <fun>

1

app 2 (+);;
-: int -> int = <fun>

1

let x::y = let _::t = [1;2;3] in t;;
val x : int = 1
val y : int list = [2;3]

1/2

List.fold_left (fun x f -> f x) 0 [(+); (-); fun x -> - x * x];;
-: int = -81

1

let rec f op = function
 (h1::h2::t) -> op h1 h2 :: f op t
 | _ -> [];;
val f : ('a -> 'a -> 'b) -> 'a list -> 'b list = <fun>

1

f max [3;2;4;5;1];;
-: int list = [3;5]

1

85/3
WUOLAH

2. (1 punto) Realice una implementación recursiva terminal, de la forma más sencilla posible, para la función $\text{drop}: \text{int} \rightarrow \text{'a list} \rightarrow \text{'a list}$, de modo que $\text{drop } n \text{ l}$ devuelva la lista que resulta al descartar los n primeros elementos de l (Por ejemplo, $\text{drop } 2 \text{['a'; 'e'; 'i'; 'o'; 'u']}$ debe ser la lista $['i'; 'o'; 'u']$)

Realice unha implementación recursiva terminal, o más sinxela posible, para a función $\text{drop}: \text{int} \rightarrow \text{'a list} \rightarrow \text{'a list}$, de xeito que $\text{drop } n \text{ l}$ devolva a lista que resulta de desbotar os n primeiros elementos de l . (Por exemplo, $\text{drop } 2 \text{['a'; 'e'; 'i'; 'o'; 'u']}$ debe ser a lista $['i'; 'o'; 'u']$)

```
let drop n l =  
    let rec aux n l = match l with  
        [] → []  
        h::t → if n > 0 then aux (n-1) t  
                else t  
    in aux n l;  
"drop n l" -0/3
```

(0/7)

3. (1,5 puntos) Realice una implementación recursiva terminal, de la forma más sencilla posible, para la función $\text{take}: \text{int} \rightarrow \text{'a list} \rightarrow \text{'a list}$, de modo que $\text{take } n \text{ l}$ devuelva la lista con los n primeros elementos de l . Si n es negativo debe provocarse la excepción `Invalid_argument "take"`. Si l no tiene suficientes elementos, debe provocarse la excepción `Failure "take"`. (Por ejemplo, $\text{take } 3 \text{['a'; 'e'; 'i'; 'o'; 'u']}$ debe ser la lista $['a'; 'e'; 'i']$)

Realice unha implementación recursiva terminal, o más sinxela posible, para a función $\text{take}: \text{int} \rightarrow \text{'a list} \rightarrow \text{'a list}$, de xeito que $\text{take } n \text{ l}$ devolva a lista cos n primeiros elementos de l . (Se n é negativo debe provocarse a excepción `Invalid_argument "take"`. Se l non ten elementos dabondo, debe provocarse a excepción `Failure "take"`). (Por exemplo $\text{take } 3 \text{['a'; 'e'; 'i'; 'o'; 'u']}$ debe ser a lista $['a'; 'e'; 'i']$)

```
let take n l =  
    if n < 0 then raise (Invalid_argument "take")  
    else if List.length l < n then raise (Failure "take")  
    else  
        let rec aux n l l1 = match l with  
            [] → list.rev l1  
            h::t → if n > 0 then aux (n-1) t (h::l1)  
                    else aux 0 l1 l1  
        in aux n l [];
```

(1/4)

list.rev l1

else
 let rec aux n l l1 = match l with
 [] → list.rev l1
 h::t → if n > 0 then aux (n-1) t (h::l1)
 else aux 0 l1 l1
 in aux n l [];

Que no te escriban poemas de amor cuando terminen la carrera

(a nosotros por suerte nos pasa)



Ayer a las 20:20

Oh Wuolah wuolitah
Tu que eres tan bonita

Siempre me has ayudado
Cuando por exámenes me he
agobiado

Llegó mi momento de despedirte
Tras años en los que has estado mi
lado.

Pero me voy a graduar.
Mañana mi diploma y título he de
pagar

No si antes decirte
Lo mucho que te voy a recordar



Envía un mensaje...



WUOLAH

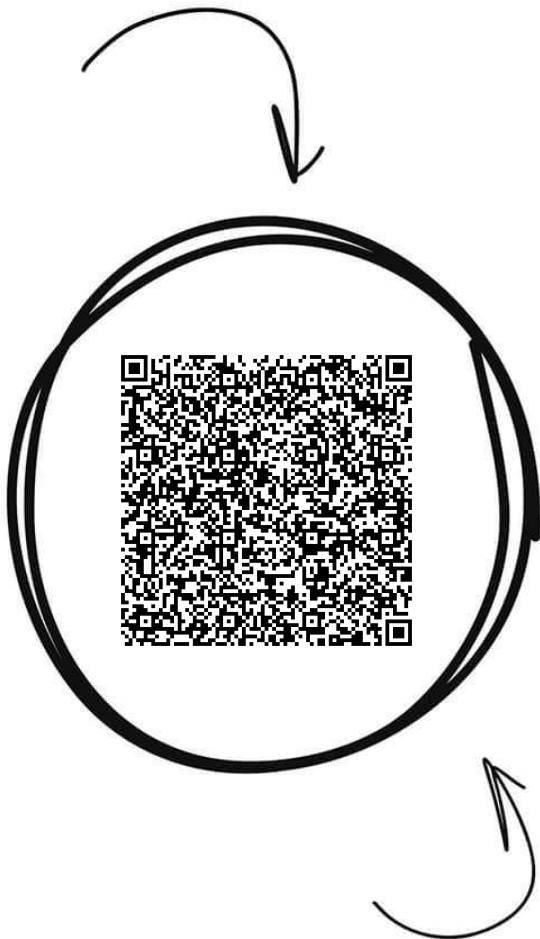


Paradigmas de Programación



Comparte estos flyers en tu clase y consigue más dinero y recompensas

- 1 Imprime esta hoja
- 2 Recorta por la mitad
- 3 Coloca en un lugar visible para que tus compis puedan escanear y acceder a apuntes
- 4 Llévate dinero por cada descarga de los documentos descargados a través de tu QR



Banco de apuntes de la



4. (1,5 puntos) Considere la función `ord` definida a continuación de modo imperativo:

Considere a función `ord` definida de seguido de xeito imperativo:

```
let ord l =  
    if l = [] then true else  
    let l = ref l in  
    while List.tl !l <> [] &&  
        List.hd !l <= List.hd (List.tl !l)  
    do  
        l := List.tl !l  
    done;  
    List.length !l = 1;;
```

¿Cuál es el tipo de `ord`?

Cal é o tipo de `ord`?

ord : $\lambda l : \text{list} \rightarrow \text{bool} = \text{<fun>}$

0'25

Redefina `ord` de modo funcional (sin usar bucles ni variables). En esta definición dé preferencia al pattern-matching sobre el `if_then_else_`.

Redefina `ord` de xeito funcional (sen empregar bucles nin variables). Nesta definición dea preferencia ao pattern-matching sobre o `if_then_else_`.

```
let rec ord l = match l with  
    [] → true  
    h1::[] → true  
    h1::h2::t → if h1 <= h2 then ord (h2::t)  
                 else false;;
```

1

¿Es recursiva terminal la implementación funcional que ha realizado para `ord`? (Indique muy brevemente por qué)

É recursiva terminal a implementación funcional que realizou para `ord`? (Sinale moi brevemente por que)

Si porque la última operación que se realiza es la llamada a `ord`, teniendo en cuenta que el `if c1 <= c2 then c2 else false` equivale a `c1 <= c2 && c2`, es decir, para que evalúe la segunda condición tiene que cumplirse la primera.

0'25

Que no te escriban poemas de amor
cuando terminen la carrera ➤➤➤➤➤



WUOLAH

(a nosotros por suerte nos pasa)

No si antes decíte
Lo mucho que te voy a recordar

Pero me voy a graduar.
Mañana mi diploma y título he de
pagar

Llegó mi momento de despedirme
Tras años en los que has estado mi
lado.

Siempre me has ayudado
Cuando por exámenes me he
agobiado

Oh Wuolah wuolah
Tu que eres tan bonita

5. (1 punto) Dado el tipo de dato '`a tree`' definido a continuación para representar árboles binarios con nodos etiquetados con valores de tipo '`a`', defina una función `inner_nodes`: '`a tree` → `a list`' que dé, para cada árbol, la lista en orden de sus nodos internos (esto es, aquellos nodos que no son hojas).
Dado o tipo de dato '`a tree`' definido de seguido para representar árboles binarias con nós etiquetados con valores de tipo '`a`', defina unha función `inner_nodes`: '`a tree` → `a list`' que dea, para cada árbore, a lista en inorden dos seus nós internos (é dicir, os nós que non son follas).

`type 'a tree = E | N of 'a tree * 'a * 'a tree`

Let rec `inner_nodes` = function
 $E \rightarrow []$
 $| N(E, r, E) \rightarrow [r]$
 $| N(i_izq, r, i_der) \rightarrow (\text{list.append}(\text{inner_nodes } i_izq) [r]) @ \text{inner_nodes } i_der; ;$

1