Examen 20 Enero 2016, preguntas

Paradigmas de Programación (Universidade da Coruña)

PARADIGMAS DE PROGRAMACIÓN

2016.01.20

APELIDOS *NOME*

1. *(1 punto) Indique o tipo (ou erro de tipo) para cada unha das seguintes expresións en OCaml.*

```
not (if true then 2 = 3 else 'a' <> 'c')
```

☐ int ☐ float ☐ char
☐ bool ☐ string ☐ type error

```
if not false then float_of_int 2 else sqrt (sqrt 16.)
```

☐ int ☐ float ☐ char
☐ bool ☐ string ☐ type error

```
if 'a' = String.get "abcd" (if 0. = 1. then 2 else 3) then Char.code 'A' else 65
```

☐ int ☐ float ☐ char
☐ bool ☐ string ☐ type error

```
1 + (if "int" <> "float" then 2. else 3.) +. 4.
```

☐ int ☐ float ☐ char
☐ bool ☐ string ☐ type error

2. *(1 punto) Indique o efecto da compilación e execución das seguintes frases en OCaml (con tipos e valores), como o indicaría o compilador interactivo. Distinga claramente expresións de definicións.*

```
let clasif p x (pos,neg) = if p x then (x::pos,neg) else (pos,x::neg);;
```

```
let divide p l = List.fold_right (clasif p) l ([],[]);;
```

```
divide (fun x -> x mod 2 = 0) [1;2;3;4;5];;
```

3. *(1 punto)* Defina **de xeito recursivo terminal** unha función `last_element:'a list -> 'a` que devolva o último elemento da lista á que se aplique. Se a lista está baleira debe provocar a excepción ***Invalid_argument "last_element"***.

4. *(1 punto)* Defina unha función (procedimento) `output_multiples: int -> int -> int -> unit` de xeito que ***output_multiples x a b*** "imprima" na saída estándar todos os múltiplos de *x* no intérvalo *[a, b]*, un por línea.

5. O tipo de dato ***'a clist***, definido a continuación, serve para representar listas de valores de tipo ***'a***. Basta con considerar que ***Empty*** representa a lista baleira, que ***Single x*** representa a lista cun único elemento con valor ***x***, e que ***App (l1,l2)*** representa a concatenación das listas ***l1*** e ***l2*** (se ***l1*** e ***l2*** representan listas do mesmo tipo).

```
type 'a clist = Empty | Single of 'a | App of 'a clist * 'a clist
```

Con esta representación, existirían distintos valores de tipo ***'a clist*** que representarían a mesma lista de valores de tipo ***'a***. Así, por exemplo, ***App (App (Single 1, Single 2), Single 3)*** representaría a mesma lista que ***App (Single 1, App (Single 2, Single 3))***; como tamén o farían ***Empty*** e ***App (Empty, Empty)***.

(0,5 puntos) Defina unha función ***is_empty: 'a clist -> bool*** que sirva para determinar se un valor deste tipo representa a lista baleira.

(0,5 puntos) Con esta representación pódese implementar a concatenación de listas de xeito que se avalíe en tempo constante. Fágao cunha función ***append : 'a clist -> 'a clist -> 'a clist***

(0,5 puntos) Defina tamén as funcións ***hd: 'a clist -> 'a*** e ***tl: 'a clist -> 'a clist*** , que sirvan para obter, respectivamente, a cabeza e a cola dunha lista. Estas funcións deben provocar excepcións ***Invalid_argument*** _ se se aplican a un argumento inadecuado.

(0,5 puntos) Defina unha función ***eq_clist : 'a clist -> 'a clist -> bool*** que sirva para determinar se dous valores de tipo ***'a clist*** representan a mesma lista.

PARADIGMAS DE PROGRAMACIÓN

XANEIRO 2016: Parte de obxetos / *ENERO 2016: Parte de objetos*

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

[0.6 puntos] EXERCICIO / *EJERCICIO 1*:

Indique cales das seguintes definicións son correctas ou non. 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 correctas o no. 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 claseA =
object (self)
  val atributo1 = 10
  method metodo1 () = 1 + atributo1
  method metodo2 () = self#metodo1() + 1
end;;
```

```
class claseA =
object
  val atributo1 = 10
  method metodo1 () = 1 + atributo1
  method metodo2 () = metodo1() + 1
end;;
```

```
class claseA =
object
  val atributo1 = 10
  method metodo1 () = 1 + atributo1
  method metodo2 () = this#metodo1() + 1
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 claseK =
object
  val atributo1 = [10; 20; 30]
  method metodo1 ()  = atributo1
  method metodo2 ()  = List.hd atributo1
  method metodo3 ()  = List.tl atributo1
end;;
```

```
class subclaseJ6 (m,n) =
object
  inherit claseJ as super
  val atributo1 = [m; n]
end;;

let j = new claseJ;;
let k = new claseK;;
let sj62 = new subclaseJ6;;
```

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; sj62 (0,0)];;
```

```
[k; sj62 (0,0)];;
```

```
k::List.tl [sj62 (0,0)];;
```

[0.4 puntos] EXERCICIO / *EJERCICIO 3*:

```
class claseR =
object (this)
  val mutable atri = [1; 2; 3; 4; 5]
  method metodo1 () = atri
  method metodo2 () = List.hd (this#metodo1())
  method metodo3 () = List.tl (this#metodo1())
end;;
```

```
class subclaseR1 pc1 =
object (this)
  inherit claseR as super
  val mutable atri = pc1
  method metodo4 pm1 = atri <- pm1::atri
end;;
```

Dadas as definicións anteriores, indique cales das seguintes funcións aceptarían como entrada unha instancia da clase `subclaseR`. No caso daquelas que a aceptarían, indique cal sería o valor obtido á saída e o seu o tipo. No caso de que as funcións non as aceptasen, explique brevemente por que.

Dadas las definiciones anteriores, indique cuáles de las siguientes funciones aceptarían como entrada una instancia de la clase `subclaseR`. En el caso de aquéllas que la aceptarían, indique cuál sería el valor obtenido a la salida y su tipo. En el caso de que las funciones no las aceptasen, explique brevemente por qué.

```
let f3 (x: <metodo1: unit->int list; metodo2: unit->int; metodo3: unit->int list>) = x#metodo2();;
```

```
let f4 (x: <metodo1: unit->int list; metodo2: unit->int; metodo3: unit->int list; ..>) = x#metodo2();;
```

[0.4 puntos] EXERCICIO / *EJERCICIO 4*:

```
class ['a,'b] claseS (pc1: 'a) =
object (this)
  val mutable arreglo = [|pc1; pc1|]
  val mutable listilla = []
  method get_arreglo = arreglo
  method get_lista: 'b list = listilla
  method set_arreglo pm1 = arreglo.(0) <- pm1; arreglo.(1) <- pm1
  method set_lista pm1 = listilla <- pm1
  method add_lista pm1 = listilla <- pm1::listilla
  method resetear_lista () = listilla <- []
end;;
```

Dada a definición anterior, e sen redefinir métodos nin atributos, extenda `claseS` nunha subclase `subclaseS1` de tal xeito que o atributo `listilla` sexa sempre de tipo lista de pares de enteiros. Indique o resultado devolto polo compilador interactivo (constructor, tipos de atributos, métodos, etc.) para tal definición.

Dada la definición anterior, y sin redefinir métodos ni atributos, extienda `claseS` en una subclase `subclaseS1` de tal forma que el atributo `listilla` sea siempre de tipo lista de pares de enteros. Indique el resultado devuelto por el compilador interactivo (constructor, tipos de atributos, métodos, etc.) para tal definición.