APELLIDOS, NOMBRE:

PROGRAMACIÓN DECLARATIVA CURSO 2019-20
Ensayo Control Programación Funcional

• Cada pregunta tiene una única respuesta correcta. Marcad con un aspa la opción elegida.

---

1. Considérense las expresiones:
```
let x = (++) in x (0:[]) [0]
let {x = [0] ++ y ; y = []} in zip x y
let x = [0]++(0:[]) in x !! 3
```
- ◯ Hay exactamente dos que están mal tipadas.
- ◯ Hay exactamente una que está mal tipada.
- ⊗ Ninguna está mal tipada.

---

2. Considérese el operador `infixl 4 ?` y las expresiones:
$e_1 =$ `f x y ? g y ? x`
$e_2 =$ `(?) (f x y) ((?) (g y) x)`
$e_3 =$ `(? x) ((? (g y)) (f x y))`
- ⊗ $e_1 \equiv e_3 \not\equiv e_2$
- ◯ $e_1 \not\equiv e_2 \not\equiv e_3 \not\equiv e_1$
- ◯ $e_1 \equiv e_2 \not\equiv e_3$

---

3. La reducción de la expresion `(\x -> (\x y -> y) x) 2` da como resultado:
- ◯ `2`
- ⊗ Una expresión de tipo `a -> a`
- ◯ Error de tipos.

---

4. Sea `f` definida por `f x y z = x y z`. El tipo de `f` es:
- ◯ `(a -> b -> c) -> b -> c -> c`
- ⊗ `(a -> b -> c) -> a -> (b -> c)`
- ◯ `a -> b -> c -> (a -> b -> c)`

---

5. Sea `f` definida por las siguientes ecuaciones:
```
f 0 y z = y
f x y z = z
```
¿Cuál de las siguientes afirmaciones es cierta?
- ◯ La función no es estricta en ninguno de sus tres argumentos.
- ◯ La función es estricta en el primer argumento y también en el segundo.
- ⊗ La función es estricta en el primer argumento pero no en el segundo.

---

6. La evaluación de `foldl (\x y -> x:y) [] [undefined, (True,False)]` da como resultado:
- ⊗ Error de tipos.
- ◯ Error de ejecución.
- ◯ Una expresión de tipo `[(Bool,Bool)]`.

---

7. Considérense las funciones f_1, f_2, f_3 definidas mediante las siguientes ecuaciones:
```
f1 x = uncurry (-) $ (if x > 0 then (x,1) else undefined)
f2 x = let g = (\(x,y) -> y - x) in (if x > 0 then g (x,1) else undefined)
f3 x = if x > 0 then ((-) x) 1 else undefined
```
- ◯ f_1 y f_2 representan la misma función.
- ◯ f_1 y f_3 no representan la misma función.
- ⊗ Las dos anteriores son falsas.

---

8. Dadas las expresiones:
```
(\x y -> y (x:[[1],[ ]]))
(\x -> (\y -> y [x,[1],[ ]]))
(\x y -> y [x,1])
```
- ◯ Las tres tienen el mismo tipo, pero no todas son equivalentes.
- ◯ Solo dos tienen el mismo tipo, pero estas dos no son equivalentes.
- ◯ Las dos anteriores son falsas.

---

9. Sea m un número natural cualquiera. La evaluación de `[take` m `(iterate (*i) i) | i <- [1..`m`]]` produce como resultado:
- ◯ Una lista de longitud $m \times m$, cuyos elementos son todas las potencias i^j, con $i, 1 \leq i \leq m$, $j, 1 \leq j \leq m$.
- ◯ Una lista de longitud m, donde el elemento i-ésimo, $1 \leq i \leq m$, es la lista de las potencias $i^1, \ldots, i^m$.
- ◯ Una lista de listas, la lista i-ésima, $1 \leq i \leq m$, tiene como elementos las potencias $1^i, \ldots, m^i$.

---

10. La expresión `zipWith (*) (filter p xs) (filter q ys)` se evalúa igual que:
- ◯ `foldr (*) (product $ filter q ys) (filter p xs)`
- ◯ `concat [[x * y | x <- filter p xs] | y <- filter q ys]`
- ◯ `foldr f [ ] (zip (filter p xs) (filter q ys)) where f (x,y) xs = (:) (x * y) xs`

---

11. La evaluación de la expresión `let {y=1:x ; x=y++[2]} in head x` produce como resultado:
- ◯ 1.
- ◯ Un error de tipo.
- ◯ Un cómputo no terminante.

---

12. ¿A cuál de las expresiones de abajo es equivalente la siguiente lista intesional?
```
[x + y | x <- [1..n], p (n-x), y <- [x..m]]
```
- ◯ `let f x = map (\y -> x + y) [x..m] in concat (map f (filter (\z -> p (n-z)) [1..n]))`
- ◯ `let f x = filter (\x -> p (n-x)) (map (\y -> x + y) [x..m]) in map f [1..n]`
- ◯ `let f x = map (\y -> x + y) [x..m] in (filter (\x -> p (n-x))) $ concat (map f [1..n])`

---

13. Considerando la definición de tipos `data T a b = A | C1 a | C2 b (T a b) deriving Show`
¿Cuántas de las siguientes expresiones están mal tipadas?
```
(C2 '2') (C1 1)            C2 '2' (C1 '1')
C2 2 (C2 1 (C1 []))        C1 (C2 'b' A)
C2 ['a'] A                 [(C1 A),C2 'a' (C2 '1' A)]
```
- ◯ Una.
- ◯ Dos.
- ◯ Ninguna.

---

14. Considerando la definición de tipos `data T a = C1 | C2 a deriving (Show, Eq, Ord)`
¿Cuántas de las siguientes expresiones evalúan a `True`?
```
compare (C2 'b') C1 == LT
compare (C2 'b') (C2 (C2 'b')) == LT
(C2 'a',C1) < (C2 'a',C2 C1)
```
- ◯ 1.
- ◯ 2.
- ◯ Ninguna.

---

15. ¿Cuál es el tipo de la expresión `do x <- getChar; putStrLn (show x++"a"); y <- getLine` ?
- ◯ `IO ()`.
- ◯ `IO String`.
- ◯ `String`.
