NOMBRE:

PROGRAMACIÓN DECLARATIVA CURSO 2016-17 PARCIALILLO 2 12-1-2017

• Cada pregunta tiene (espero) una y solo una respuesta correcta. Marcad con un aspa la opción elegida.
• **Cada respuesta correcta suma un punto; cada respuesta incorrecta resta medio punto**; las respuestas en blanco ni suman ni restan. Estad ojo avizor y suerte. Está prohibidísimo copiar.

1. Considérense las expresiones
`[[1,2]]` `([1]:[[2]]):[]` `(1:[2]):[]` `[1,2]:[]` `[1]:[2]:[]`
¿Cuál de las siguientes afirmaciones es cierta?
⊗ La primera, la tercera y al menos otra más son sintácticamente equivalentes entre sí
◯ La segunda, la cuarta y al menos otra más son sintácticamente equivalentes entre sí
◯ Las dos anteriores son falsas.

---

2. Considérense las expresiones de tipo (que solo difieren en los paréntesis): τ_1 = `(a -> a) -> (a -> a) -> (a -> a)`
τ_2 = `(a -> a) -> (a -> a) -> a -> a`
τ_3 = `(a -> a) -> ((a -> a) -> (a -> a))`
◯ $\tau_1 \not\equiv \tau_2 \not\equiv \tau_3 \not\equiv \tau_1$ ◯ $\tau_1 \equiv \tau_3 \not\equiv \tau_2$ ⊗ $\tau_1 \equiv \tau_2 \equiv \tau_3$

---

3. Considérense las expresiones (que solo difieren en los paréntesis): e_1 = `f x + g z 4`
e_2 = `f x ((+) g (z 4))`
e_3 = `(+) (f x (g z 4)`
⊗ $e_1 \not\equiv e_2 \not\equiv e_3 \not\equiv e_1$ ◯ $e_1 \equiv e_2 \equiv e_3$ ◯ $e_1 \equiv e_3 \not\equiv e_2$

---

4. La evaluación de la expresión
`foldr (\x y -> not x || y) False [False,True,undefined]` da como resultado
⊗ `True` ◯ `False` ◯ Un error en tiempo de ejecución

---

5. La evaluación de la expresión
`foldl (\x y -> not x || y) False [False,True,undefined]` da como resultado
◯ `True` ◯ `False` ⊗ Un error en tiempo de ejecución

---

6. Sea l = `[(x,y)|x <- [2,4],y <- [1..x],b]`, donde b es una cierta expresión booleana , y sea n =`length` l.
¿Cuántas de la siguientes situaciones: $n = 1$ $n = 5$ $n = 7$ pueden darse?
◯ Solo una de ellas ⊗ Solo dos de ellas ◯ Las tres son posibles

---

7. La reducción de la expresión `(\x y -> y (y x)) 1 (\x -> x+2)` producirá el resultado
◯ 3 ⊗ 5 ◯ 7

---

8. Sea `f` definida por `f x y = y (y x)`. El tipo de `f` es:
⊗ `a -> (a -> a) -> a`
◯ `(a -> a) -> a -> a`
◯ `f` está mal tipada

---

9. Considérense las siguientes expresiones:
```
take 30 (reverse [1..10^30])
reverse (take 30 [1..10^30])
last (takeWhile (< 1000) (iterate (+ 2) 1))
```
¿Cuántas de ellas nos llevará toda la vida evaluarlas?
⊗ Exactamente una de ellas ◯ Exactamente dos de ellas ◯ Las tres

---

10. Sea `f` definida por las siguientes ecuaciones:
```
f  x  False  = x
f  y   True  = not y
```
¿Cuál de las siguientes afirmaciones es cierta?

⊗ La función es estricta en sus dos argumentos
○ La función es estricta en el segundo pero no en el primer argumento
○ Las dos anteriores son falsas.

---

11. ¿Cuál de los siguientes tipos para la expresión e hace que la expresión
`zipWith` e `(iterate not True) (iterate (+ 1) 0)` esté bien tipada?

○ `[Bool] -> [Int] -> [Bool]`
○ `[Bool] -> [Int] -> [(Bool,Int)]`
⊗ `Bool -> Int -> Char`

---

12. La evaluación de `(head.(!! 1)) (map (zip [0..3]) [[1..4],[2..5]])` produce como resultado

○ `1` ○ `(1,2)` ⊗ `(0,2)`

---

13. Considérense las expresiones siguientes:

$e_1 \equiv$ `\x -> ((\y -> x) x)`
$e_2 \equiv$ `\x -> ((\y -> x+y) y)`
$e_3 \equiv$ `let y=[1,2,3] in let x= y!!1 in x*head y`
$e_4 \equiv$ `let {y=2*x;x=5} in y*y*x`
$e_5 \equiv$ `[i+j | i<-[1..100],j<-[0..i],mod j i == 0]`

¿Cuántas de ellas son sintácticamente erróneas por problemas de ámbito de variables?

⊗ Exactamente una de ellas ○ Tres o más de ellas ○ Las dos anteriores son falsas.

---

14. La evaluación de la expresión `let x= 1:3:map (+ 1) x in last (take 3 x)` produce como resultado:

⊗ `2`
○ Un error en ejecución
○ Un error sintáctico o de tipos

---

15. ¿Cuántas de las siguientes definiciones de tipos (independientes unas de otras) son correctas?
```
data Tip    = A  | C Int Tip  | A (Int,Int,Tip)
data Tap    = A  | C Int Tap  | (Int,Int,Tap)
data Top a  = A  | C a        | D a b
```
○ Una de las tres ○ Dos de las tres ⊗ Ninguna de las tres

---

16. ¿Cuáles de las dos siguientes expresiones representa correctamente una acción de I/0?
```
do  x <- getChar          do  x <- getChar
    x                         return x
```
○ Las dos ○ Solo la primera ⊗ Las dos anteriores son falsas.

---

17. El tipo que inferirá Haskell, teniendo en cuenta clases de tipos, para una función `f` definida por
`f x y z = if x <= y+1 then z else z+1` será:

⊗ `f :: (Num a, Ord a, Num b) => a -> a -> b -> b`
○ `f :: (Ord a, Num a) => a -> a -> a -> a`
○ `f :: (Ord a, Num b) => a -> b -> b -> b`

---

18. Considérense la declaraciones de clase e instancia
```
class C a where  f, g:: a -> Int   | instance C Bool where  f x = if x then 0 else 1  | instance C Int where  g x = x
                 f x = g x + 1     |                                                   |
```
¿Qué afirmación es correcta?

Nota: *Se dan dos soluciones como correctas: la primera es acorde a lo visto directamente en clase; la segunda es acorde a alguna respuesta marcada como correcta en la colección de test previos, y corresponde a versiones anteriores de Hasjell.*

⊗ `f 0 + g 0` se evalúa a `1` y `f True` se evalúa a `0`
○ `f True` se evalúa a `2`
⊗ Las dos anteriores son falsas.
