APELLIDOS, NOMBRE:

PROGRAMACIÓN DECLARATIVA CURSO 2019-20
Control Programación Funcional

• Cada pregunta tiene una única 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.

---

1. Considérense las siguientes expresiones:
```
let x = (:) in x ( x 0 []) []
let {x = (:) 0 ; y = []} in zip (x y) y
let x = (0:[]) in x !! 2
```
◯ Hay exactamente dos que están mal tipadas.
◯ Hay exactamente una que está mal tipada.
⊗ Ninguna está mal tipada.

---

2. Dadas las siguientes expresiones: e_1 = `(+ y) ((f x) (y ^ x))`
e_2 = `(f x (( ^ ) y x) ) + y`
e_3 = `((f x) ( y^ ) x) + y`
◯ $e_1 \not\equiv e_2 \not\equiv e_3 \not\equiv e_1$
◯ $e_1 \equiv e_3 \not\equiv e_2$
⊗ $e_1 \equiv e_2 \not\equiv e_3$

---

3. La reducción de la expresion `(\x y -> (\x -> y) x) 2` es equivalente a:
◯ `2`
◯ `\x -> 2`
⊗ `\x -> x`

---

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

---

5. Sea `f` definida por las siguientes ecuaciones:
```
f x y True = y
f 1 y z = y
f x y z = z
```
¿Cuál de las siguientes afirmaciones es cierta?
◯ La función es estricta en sus tres argumentos.
⊗ La función es estricta solo en el tercer argumento.
◯ Las dos anteriores son falsas.

---

6. La evaluación de `(map head (iterate tail [1..])) !! 1` da como resultado:
◯ `1`
⊗ `2`
◯ No termina.

---

7. ¿Cuántas de las tres siguientes definiciones de tipos son correctas?
```
data T1 a b   = C1 b           | C2 (T1 a)
data T2 a     = C1' Int        | C2' (Int,Bool) a
data T3 a     = C1'' Int Bool  | C2'' a
```
◯ Las tres.
◯ Exáctamente una de las tres.
⊗ Exáctamente dos de las tres.

---

8. El resultado de evaluar la expresión

```
let f x = let g = (\(x,y) -> y - x) in (if x > 0 then g (x, 1) else undefined) in f 1
```

da como resultado:

◯ Error de ejecución.
◯ No se puede evaluar porque está mal tipada.
⊗ Las dos anteriores son falsas.

---

9. El tipo de `(\x y -> drop x)` es:

⊗ `Int -> b -> [a] -> [a]`
◯ `Int -> Int -> [a] -> [a]`
◯ Está mal tipada.

---

10. La evaluación de `[take j [3..i] | i <- [1..4], i > 2, j <- [i-1,i]]` produce como resultado:

◯ Una lista de números de longitud 4, siendo los dos primeros iguales entre sí.
◯ Una lista de listas de números, siendo las dos primeras listas vacías.
⊗ Una lista, cuyos dos últimos elementos son iguales.

---

11. La expresión:

```
foldr f [ ] (zip (filter p xs) (filter q ys)) where f (x,y) xs = (:) (x * y) xs
```

se evalúa igual que:

◯ `foldr (*) (product $ filter q ys) (filter p xs)`
⊗ `zipWith (*) (filter p xs) (filter q ys)`
◯ `concat [[x * y | x <- filter p xs] | y <- filter q ys]`

---

12. La evaluación de `foldl (\x y -> x++y) [] [[],undefined]` da como resultado:

◯ Error de tipos.
⊗ Error de ejecución.
◯ Una expresión de tipo `[a]`.

---

13. ¿A cuál de las expresiones de abajo es equivalente la siguiente lista intensional?

```
[(x,y) | x <- [1..n], p x, y <- [1..m], q y]
```

⊗ `concat (map f (filter p [1..n])) where f z = map (\y -> (z, y)) (filter q [1..m])`
◯ `map f (filter p [1..n]) where f x = map (\y -> (x, y)) (filter q [x..m])`
◯ `filter p $ concat (map f [1..n]) where f x = map (\y -> (x,y)) (filter q [1..m])`

---

14. Considerando la definición de tipos

```
data   T a = P | Q a (T a)
       deriving (Show, Eq, Ord)
```

¿Cuántas de las siguientes expresiones son sintácticamente correctas?

```
Q [ ] P     Q P P            [Q 1 (Q 1 P),P]
Q P (Q P)   Q [ ](Q [P] P)   (P, P 1)
```

◯ Todas menos una.
⊗ Todas menos dos.
◯ Todas menos tres.
