

Músicos - 2025 - Parcial Funcional

Necesitamos modelar requerimientos para una importante personalidad del ámbito musical, que nos contó que un músico tiene un nombre, el grado de experiencia (un número que va de 1 a 10), su instrumento favorito y por último el historial de actuaciones que consiste en una lista con la fecha (día, mes y año) y la cantidad de público que asistió.



Punto 1: Informer

En este punto no puede utilizar funciones auxiliares, solo composición y aplicación parcial. No puede utilizar recursividad.

- Queremos saber si un músico tuvo actuaciones donde hubo más de 5.000 asistentes
- Queremos saber cuántas actuaciones tuvo un músico en un año en particular

Punto 2: Radioactivity

Existen varias actividades que pueden hacer los músicos

- Tocar un instrumento: le cambia el instrumento favorito al músico y además le suma 1 a su experiencia si el instrumento es "oboe", "fagot" o "cello"
- Cantar, que no cambia el instrumento principal pero le agrega como prefijo "Lalala" al nombre porque lo pone contento
- Hacer una presentación, que le agrega una presentación al historial del músico para una fecha y un público, obviamente al final de esa lista. Eso le suma 1 a su experiencia y si el músico toca el "oboe" o el "fagot" o el "cello", suma 1 punto más de experiencia
- Pensar, que es una actividad muy importante, pero que no produce cambios en el músico

Indicar cómo podemos hacer que un músico

- toque el fagot
- cante
- haga una presentación para el día 2 de agosto de este año con 300 asistentes
- piense

Considere que el músico tenga inicialmente una experiencia de 5 y al piano como su instrumento favorito.

Punto 3: Mucha experiencia

Queremos saber cuánta experiencia le hace ganar a un músico una lista de actividades. Por ejemplo: tocar el oboe, cantar y hacer una presentación para 500 personas debería darle 3 puntos extra de experiencia a los que ya tenía el músico.

En este punto no puede utilizar recursividad.

Punto 4: Obsesionario en la mayor

Queremos saber si un músico tiene presentaciones correctas, esto significa que la cantidad de público en las presentaciones que están en la posición impar es impar, y la cantidad de público en las presentaciones que están en la posición par son pares.

Algunos ejemplos:

- sin presentaciones, el músico tiene presentaciones correctas
- si la secuencia de presentaciones es 501, 2000, 755 el músico tiene presentaciones correctas
- si la secuencia de presentaciones es 501, 2000, 755, 1000 el músico tiene presentaciones correctas
- si la secuencia de presentaciones es 501, 2001, 755 el músico no tiene presentaciones correctas

En este punto debe utilizar recursividad.

~~XXXXXX~~
data Musico = Musico {
 nombre :: String,
 gradoXP :: Int,
 instrumento :: String,
 actuaciones :: [Actuacion]
}

9 (Nueve)

data Actuacion = Actuacion {
 fecha :: (Int, Int, Int),
 cantidad :: Int
}

a) masDe5000 :: Musico -> Bool
 masDe500 = any (>5000). map cantidad. actuaciones
 ↳ Es poco declarativo

b) cuantas Actuaciones En :: Int -> Musico -> Int
 cuantas Actuaciones En año musico =
 length. filter (== año). map año. map fecha. actuaciones \$ musico
 ↳ Es poco declarativo. Delegar

año :: (a, a, a) -> a
año (-, -, año) = año

a) tocar Instrumento :: String -> Actividad
 tocar Instrumento instrumentazo musico =
 | elem instrumentazo especificos = map GradoXP (+1). map Instrumento (const instrumentazo) \$ musico
 | otherwise = map Instrumento (const instrumentazo) \$ musico
 ↳ Delegar

especificos :: [String]
especificos = ["dba", "fagot", "cello"]

map Instrumento :: (String -> String) -> Musico -> Musico
map Instrumento f musico = musico { instrumento = f instrumento \$ musico }
 ↳ Hace falta ser más expresivo

map GradoXP :: (Int -> Int) -> Musico -> Musico
map GradoXP f musico = musico { gradoXP = f gradoXP \$ musico }

type Actividad = Musico -> Musico

cantar :: Actividad

cantar = map Instrumento (: "Lalala")

hacerPresentacion :: Activación → Actividad

hacerPresentacion activacion musico =

siToca. map Grado XP (+1). map Actuaciones (++ [.: activacion])

siToca :: TMusico → TMusico

siToca musico

1 flip elem especificas. instrumento \$ musico = map Grado XP (+1)

otherwise = id TMusico

map Actuaciones :: ([Activación] → [Activación]) → TMusico → TMusico

map Actuaciones f musico = musico { actuaciones = f. actuaciones \$ musico }

pensar :: Actividad

pensar = id

actividadEjemplo :: Actividad

actividadEjemplo =

pensar. hacerPresentacion (Actividad (2.8.2025) 300). cantar. tocar Instrumento "fagot"

musicoEjemplo :: TMusico

musicoEjemplo = Musico "charly" 5 "Piano" []

> actividadEjemplo musicoEjemplo

Musico "charly" 8 "lalala fagot" [Actividad (2.8.2025) 300]

3) cuentaExperiencia :: Musico → [Actividad] → Int

cuentaExperiencia musico actividades =

gradoXP. foldL (\musico actividad → actividad musico) musico actividades

4) tienePresentacionesCorrectas :: Musico → Bool

tienePresentacionesCorrectas musico = evaluar . map cantidad . actuaciones \$ musico

evaluar :: [Int] → Bool

evaluar [] = True

evaluar [pos1] = False → no siempre. depende si a pos1 es impar o par

evaluar (pos1 : pos2 : tail) =

~~((not even \$ pos1) && (even \$ pos2)) && evaluar (pos2 : tail)~~ ^{odd} ^{bien}

9 (Nueve)

Hoja 2/2

Máximo Torado

Docente: Raulo Gonzalez