Programación Funcional

Aclaraciones:

- *Esta evaluación es a libro abierto. Se pueden usar todas las funciones y propiedades vistas en clase, aclarando la referencia. Cualquier otra función o propiedad que se utilice* ***debe ser definida o demostrada****.*
- *No se olvide de poner nombre, nro. de alumno, nro. de hoja y cantidad total de hojas en cada una de las hojas.*
- *Le recomendamos leer el enunciado en su totalidad y organizar sus ideas antes de comenzar la resolución.*
- *Recuerde que reusar código es una forma muy eficiente de disminuir el tiempo necesario para programar.*
- *La intención de la evaluación es medir cuánto comprende usted del tema. Por ello, no dude en escribir todo lo que sabe, explicar lo que se propone antes de escribir código y probar sus funciones con ejemplos.*

Las construcciones espaciales grandes son ensambladas en el espacio mediante el acoplamiento de distintos módulos, que son puestos en órbita de forma independiente, dado que es inviable poner en órbita la construcción completa. Considere la siguiente representación para naves espaciales, donde cada módulo alberga un único componente y se le pueden conectar otros dos módulos:

```
data Component = Cargo | Engine | Shield | Cannon
data Spaceship = Module Component Spaceship Spaceship | Plug
```

Distinguimos al nodo raíz que es desde donde se tripula la nave (independientemente del componente adjunto) y consideraremos los siguientes tipos de componentes:

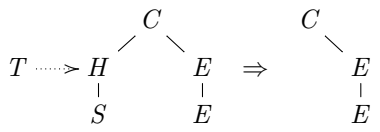
- Espacios de carga (H), cada uno aportando una unidad de almacenamiento.
- Motores (E) que propulsan la nave, cada motor ubicado en un nodo hoja provee una unidad de empuje.
- Generadores de campos de fuerza (S) que protegen a la nave de colisiones con objetos pequeños.
- Cañones de plasma (C) que sirven para evitar colisiones con objetos grandes convirtiéndolos en pequeños.

Consideraremos que los peligros encontrados en el espacio pueden provenir de la izquierda o la derecha de la cabina, pero nunca del frente o de atrás. Por lo que los modelaremos de la siguiente forma:

```
data Direction = Larboard | Starboard
data Size   = Small | Big | Torpedo
type Hazard = (Direction, Int, Size)
```

El entero asociado a un peligro indica a qué nivel con respecto a la cabina impactará un objeto. Identificamos tres tipos de peligros, objetos pequeños, grandes y torpedos. Estos últimos no pueden ser evadidos por las naves, por lo que siempre producen un impacto.

Ejemplo:



La nave recibe un impacto de un torpedo por la izquierda a nivel 2 y todos los componentes dependientes del nodo impactado se desprenden.

Ejercicio 1

Escriba las siguientes funciones:

a) `shielded :: Spaceship -> Bool`
que indica si la nave posee al menos un generador de campos de fuerza.

b) `armed :: Spaceship -> Bool`
que indica si la nave posee al menos un cañón.

c) `thrust :: Spaceship -> Int`
que retorna el poder de propulsión de una nave.

d) `wreck :: Hazard -> Spaceship -> Spaceship`
que devuelve la nave resultante de desprender los módulos dependientes del módulo donde se recibe un impacto (esta función asume que se produce el impacto).

Ejercicio 2

De el tipo y una implementación para una función que generalice la recursión sobre las naves espaciales (`foldSS`).

Ejercicio 3

Escriba las siguientes funciones sin usar recursión explícita, es decir, usando esquemas (foldSS, foldr, etc). Además puede usar las funciones del ejercicio 1 sin redefinirlas.

a) `capacity :: Spaceship -> Int`
que retorna la capacidad de la nave, donde cada módulo de carga aporta una unidad de capacidad.

b) `largest :: [Spaceship] -> Spaceship`
que dada una lista de naves retorna una de capacidad máxima.

c) `dimensions :: Spaceship -> (Int,Int)`
que dada una nave retorna su alto y ancho (pensando el alto como la cantidad de componentes de la rama más larga y el ancho como la cantidad de componentes del nivel más ancho).

d) `manoeuvre :: Spaceship -> [Hazard] -> Spaceship`
que simula el resultado de maniobrar una nave a través de una serie de peligros. Si se encuentra un objeto pequeño y la nave está escudada no se produce impacto. Si el objeto es grande y la nave está armada entonces se transforma en un objeto pequeño. Si es un torpedo no se puede evitar el impacto.

e) `test :: [Spaceship] -> [Hazard] -> [Spaceship]`
que dada una lista de naves y una lista de peligros retorna la lista de naves que sobreviven los peligros (es decir las naves con motores funcionales luego de navegar a través de los meteoros).

Ejercicio 4

Dadas las siguientes funciones:

```
components :: Spaceship -> [Component]
components Plug             = []
components (Module c s1 s2) = components s1 ++ [c] ++ components s2

replace :: (Component -> Component) -> Spaceship -> Spaceship
replace f Plug             = Plug
replace f (Module c s1 s2) = Module (f c) (replace f s1) (replace f s2)
```

Demuestre que: $\forall$`sp:Spaceship` . `componentes (replace f sp) = map f (componentes sp)`