Parcial 1 Estructuras de Datos 2017s2

Aclaraciones:

- *Esta evaluación es a libro abierto. Se pueden usar todas las funciones vistas en clase, aclarando la referencia. Cualquier otra función que utilice debe ser definida.*
- *No se olvide de poner nombre, nro. de hoja, y cantidad total de hojas en cada una de las hojas.*
- ***Deje la primera media página en blanco**, para que podamos incluir ahí las correcciones.*
- *Le recomendamos leer el enunciado en su totalidad antes de comenzar.*

Definimos un tipo de datos `Nave` que está construido recursivamente (una nave está compuesta por más partes de nave, como en el ejemplo de pizzas visto en clase). Este tipo recursivo tiene forma de árbol.

```
data Contenedor = Comida | Oxigeno | Torpedo | Combustible
data Componente = Escudo | CañonLaser | Lanzatorpedos | Motor Int | Almacen [Contenedor]
data Nave       = Parte Componente Nave Nave | ParteBase
```

Una nave puede estar compuesta por una parte base sin componentes, o por varios componentes hasta llegar a partes base. Además un motor tiene un poder de propulsión y un almacén cuenta con una lista de contenedores.

Ejercicio 1 Construya un valor de tipo `Nave` que contenga un motor, un escudo, dos armas y un almacen que posea comida.

Ejercicio 2

Resuelva las siguientes funciones, **con recursión estructural sobre el tipo Nave**:

a) `componentes :: Nave -> [Componente]`
Retorna la lista de componentes.
Ejemplo:

```
componentes (Parte Escudo ParteBase (Parte Lanzatorpedos ParteBase ParteBase)
=> [Escudo, Lanzatorpedos]
```

b) `poderDePropulsion :: Nave -> Int`
Retorna el poder de propulsión de una nave. El poder de propulsión de una nave es la suma de los poderes de propulsión de los motores de la nave.
Ejemplo:

```
poderDePropulsion (Parte Lanzatorpedos
                (Parte (Motor 5) ParteBase ParteBase)
                (Parte (Motor 20) ParteBase ParteBase)
=> 25
```

c) `desarmarse :: Nave -> Nave`
Reemplaza armas por escudos.
Ejemplo:

```
desarmarse (Parte Lanzatorpedos
                (Parte CañonLaser ParteBase ParteBase)
                (Parte (Motor 20) ParteBase ParteBase)
=>
(Parte Escudo
    (Parte Escudo ParteBase ParteBase)
    (Parte (Motor 20) ParteBase ParteBase)
```

d) `cantidadComida :: Nave -> Int`
Dada una nave devuelve la cantidad de comida. Cada aparición de `Comida` vale 1.
Ejemplo:

```
cantidadComida (Parte Lanzatorpedos
                     (Parte (Almacen [Combustible, Comida, Comida]) ParteBase ParteBase)
                     (Parte (Almacen [Torpedo, Comida, Oxigeno]) ParteBase ParteBase)
=> 3
```

e) `naveToTree :: Nave -> Tree Componente`
Dada una nave la transforma en un árbol de componentes
Ejemplo:

```
naveToTree (Parte Lanzatorpedos
                ParteBase
                (Parte (Motor 20) ParteBase ParteBase)
=>
(NodeT Lanzatorpedos
    EmptyT
    (NodeT (Motor 20) EmptyT EmptyT)
```

f) `aprovisionados :: [Contendor] -> Nave -> Bool`
Dada una lista de contenedor chequea que cada almacén contenga todos esos tipos de contenedores.
Ejemplos:

```
aprovisionados [Combustible, Oxigeno, Comida]
    (Parte (Almacen [Combustible, Comida, Comida, Oxigeno, Torpedo])
         (Parte (Almacen [Torpedo, Oxigeno, Comida, Combustible]) ParteBase ParteBase)
         (Parte (Motor 20) ParteBase ParteBase)
=> True

aprovisionados [Combustible, Oxigeno, Comida]
    (Parte (Almacen [Combustible, Oxigeno, Comida])
         (Parte (Almacen [Torpedo, Torpedo]) ParteBase ParteBase)
         (Parte (Motor 20) ParteBase ParteBase)
=> False
```

g) `armasNivelN :: Int -> Nave -> [Componente]`
Devuelve las armas que haya en el nivel "n" de la nave.
Ejemplos:

```
armasNivelN 0 (Parte Lanzatorpedos
                (Parte CañonLaser ParteBase ParteBase)
                (Parte (Motor 10) ParteBase ParteBase)
=> [Lanzatorpedos]

armasNivelN 1 (Parte Lanzatorpedos
                (Parte CañonLaser ParteBase ParteBase)
                (Parte (Motor 10) ParteBase ParteBase)
=> [CañonLaser]
```