UBA – Facultad de Ciencias Exactas y Naturales – Departamento de Computación– Algoritmos y Estructura de Datos I
Parcial de Programación Funcional– 20/05/2013

LU:
Apellidos:
Nombres:

Aclaraciones: El parcial NO es a libro abierto. Cualquier decisión de interpretación que se tome debe ser aclarada y justificada. Para aprobar se requieren al menos 60 puntos. Entregar cada ejercicio en hoja separada. No está permitido utilizar alto orden. Al igual que para el TP, para la resolución del parcial, se pueden usar únicamente las funciones y operadores `last`, `init`, `head`, `tail`, `!!`, `reverse`, `++`, `elem`, `length` y los operadores de comparación entre elementos de un mismo tipo.

En esta ocasión, nos centraremos en una banda musical de reconocida barra brava de fútbol. Esta banda está compuesta por músicos de sobrada experiencia ... no precisamente musical. La banda sólo sabe/puede interpretar una sola pieza musical. Cada músico tiene su propia partitura (anotada en una servilleta de choripan usada). Las partituras se encuentran compuestas por una lista de duplas. Cada dupla representa la nota que se debe ejecutar y en qué momento. Todas las notas tienen la misma duración, duran sólo ese momento.

De esta manera, obtenemos la siguiente representación:

tipo Nombre = String;
tipo Momento = $\mathbb{Z}$;
tipo NotaMusical = Do, Re, Mi, Fa, Sol, La, Si;
tipo Partitura = [(Momento, NotaMusical)];

tipo `Musico` {
  observador `apodo` (m : Musico) : Nombre;
  observador `servilleta` (m : Musico) : Partitura;
  invariante `mtoPositivo` : $(\forall p \leftarrow servilleta(m))prm(p) \geq 0$;
  invariante $mtoOrdenadoYSinRepe(servilleta(m))$;
}

tipo `Banda` {
  observador `integrantes` (b : Banda) : [Musico];
  invariante `hayEquipo` : $|integrantes(b)| > 0$;
  invariante `musicosNoRepetidos` : $sinRepetidos(apodos(b))$;
  invariante `alguienToca` :
    $|momentosDeLosIntegrantes(b)| > 0$;
  invariante `noMeCallo` : $(\forall m1 \leftarrow rangoMomentos(b))$
    $(\exists m2 \leftarrow momentosDeLosIntegrantes(b))m1 == m2$;
}

aux `minimoMomento` (b:Banda) : Momento =
  $[m1|m1 \leftarrow momentosDeLosIntegrantes(b), (\forall m2 \leftarrow momentosDeLosIntegrantes(b))m1 \leq m2]_0$;
aux `maximoMomento` (b:Banda) : Momento =
  $[m1|m1 \leftarrow momentosDeLosIntegrantes(b), (\forall m2 \leftarrow momentosDeLosIntegrantes(b))m1 \geq m2]_0$;
aux `apodos` (b:Banda) : [Nombre] = $[apodo(i)|i \leftarrow integrantes(b)]$;
aux `rangoMomentos` (b:Banda) : [Momento] = $[minimoMomento(b)..maximoMomento(b)]$;
aux `sinRepetidos` (l : [T]) : Bool = $(\forall i \leftarrow [0..|l|))\, l[i] \notin l[(i+1)..|l|)$;
aux `primeros` (l : [(A,B)]) : [A] = $[prm(x)|x \leftarrow l]$;
aux `momentosDeLosIntegrantes` (b:Banda) : [Momento] = $[prm(p)|i \leftarrow integrantes(b), p \leftarrow servilleta(i)]$;
aux `mtoOrdenadoYSinRepe` (xs : Partitura) : Bool = $(\forall i \leftarrow [0..|xs|-1))\, prm(xs_i) < prm(xs_{i+1})$;

Las funciones que implementan estos tipos en Haskell son `apodoM :: Musico -> Nombre`, `servilletaM :: Musico -> [(Momento,NotaMusical)]`, `integrantesB :: Banda -> [Musico]`

Ejercicio 1. [45 puntos] Implementar en Haskell los siguientes problemas especificados más adelante

a) [15 p.] **problema** `piezaMusical` (b : Banda) = `result` : [(Momento, [NotaMusical])]
b) [15 p.] **problema** `soloMusical` (b : Banda, i : Musico) = `result` : Partitura
c) [15 p.] **problema** `jefeDeLaBarra` (b:Banda) = `result` : Musico

problema `piezaMusical` (b : Banda) = `result` : [(Momento, [NotaMusical])] {
  asegura $|result| == |rangoMomentos(b)|$;
  asegura $(\forall i \leftarrow rangoMomentos(b))prm(result_{i-minimoMomento(b)}) == i$;
  asegura $(\forall i \leftarrow rangoMomentos(b))mismosConjuntos(sgd(result_{i-minimoMomento(b)}), todasLasNotasDelMomento(b,i))$;
  asegura $(\forall i \leftarrow rangoMomentos(b))sinRepetidos(sgd(result_{i-minimoMomento(b)}))$;
}

aux `mismosConjuntos` (xs1 : [T], xs2 : [T]) : Bool = $((\forall x1 \leftarrow xs1)x1 \in xs2) \wedge ((\forall x2 \leftarrow xs2)x2 \in xs1)$;
aux `todasLasNotasDelMomento` (b : Banda, m : Momento) : [NotaMusical] =
  $[sgd(p)|i \leftarrow integrantes(b), p \leftarrow servilleta(i), prm(p) == m]$;

problema `soloMusical` (b : Banda, i : Musico) = `result` : Partitura {
  requiere `estaEnLaBanda` : $i \in integrantes(b)$;
  asegura $(\forall x \leftarrow result)\, tocaSolo(prm(x), i, b) \wedge notaEnMomento(prm(x), servilleta(i)) == sgd(x)$;
  asegura $(\forall m \leftarrow momentosDeLosIntegrantes(b))$
    $tocaSolo(m, i, b) \Rightarrow estaTocando(m, result) \wedge notaEnMomento(m, result) == notaEnMomento(m, servilleta(i))$;
  asegura $mtoQrdenadoYSinRepe(result)$;
}

aux `estaTocando` (m : Momento, p : Partitura) : Bool = $|[x|x \leftarrow p, prm(x) == m]| > 0$;
aux `notaEnMomento` (m : Momento, p : Partitura) : NotaMusical = $[sgd(x)|x \leftarrow p, prm(x) == m]_0$;
aux `tocaSolo` (m : Momento, i : Musico, b : Banda) : Bool =
  $estaTocando(m, servilleta(i)) \wedge |todasLasNotasDelMomento(b,m)| == 1$;

```
problema jefeDeLaBarra (b : Banda) = result : Musico {
  asegura result ∈ integrantes(b);
  asegura (∀i ← integrantes(b))cantidadDeSolos(b, result) ≥ cantidadDeSolos(b, i);
}
aux cantidadDeSolos (b : Banda, i : Musico) : ℤ = |[x|x ← rangoMomentos(b), tocaSolo(x, i, b)]|;
```

Tipos algebraicos. ⇒ Nota Importante: No está permitido CONVERTIR EL TIPO ALGEBRAICO AL TIPO LISTAS para resolver los ejercicios de tipos algebraicos (Ejercicios 2 y 3).

Se cuenta con el tipo compuesto MacSandwichito que modela las hamburguesas de una reconocida cadena de comidas rápidas.

```
tipo Ingrediente = Tomate, Jamon, Queso, Hamburguesa, Aderezo, Pan;
tipo MacSandwichito {
  observador pisos (m: MacSandwichito) : [Ingrediente];
  invariante bienFormado(m);
}
aux bienFormado (m:MacSandwichito) : Bool = tieneAlgo(m) ∧ tapasDePan(m) ∧ sinPanEnElMedio(m);
aux tieneAlgo (m:MacSandwichito) : Bool = |pisos(m)| ≥ 3;
aux tapasDePan (m:MacSandwichito) : Bool = pisos(m)_0 == Pan ∧ pisos(m)_{|pisos(m)|-1} == Pan;
aux sinPanEnElMedio (m:MacSandwichito) : Bool = (∀i ← (0..|pisos(m) - 1|))pisos(m)_i ≠ Pan;
```

donde el observador pisos(m) devuelve las lista de ingredientes de esa hamburguesa, en el orden en que se encuentran dispuestos los mismos.

Se busca implementar en Haskell algunos problemas sobre el tipo compuesto MacSandwichito mediante los tipos algebraicos `Ingrediente` y `MacSandwichito`. Dichos tipos se definen a través de los siguientes constructores:

```
data Ingrediente = Tomate | Jamon | Queso | Hamburguesa | Aderezo | Pan deriving (Eq)
data MacSandwichito = Caja | MC Ingrediente MacSandwichito deriving (Eq)
```

donde en el tipo MacSandwichito, el primer constructor permite construir un nuevo sandwich y el segundo agregarle un ingrediente al final.
Por ejemplo, un MacSandwichito m con queso tradicional, cuyos $pisos(m) == [Pan, Aderezo, Hamburguesa, Queso, Pan]$ se construiría de la siguiente manera: `MC Pan (MC Queso (MC Hamburguesa (MC Aderezo (MC Pan Caja))))`.

Ejercicio 2. [20 puntos] Implementar `uyParejaVegetariana:: MacSandwichito -> (MacSandwichito,MacSandwichito)`, que a partir de un MacSandwichito, devuelve dos sandwichitos similares al original. El primero contiene los ingredientes del original, pero sólo con sus ingredientes vegetarianos (Tomate, Queso y Aderezo). El segundo, también contiene los ingredientes del original, pero sólo con sus ingredientes cárnicos (Jamón y Hamburguesa). En los dos casos se respeta el orden original, pero con una particularidad: cuando se cuenta con dos ingredientes o más del mismo tipo consecutivos, se deja solo uno de ellos.

Por ejemplo:

```
uyParejaVegetariana
(MC Pan (MC Tomate (MC Hamburguesa (MC Hamburguesa (MC Queso (MC Tomate (MC Tomate (MC Pan Caja))))))))
```

debería dar como resultado

```
(MC Pan (MC Tomate (MC Queso (MC Tomate (MC Pan Caja)))), MC Pan (MC Hamburguesa (MC Pan Caja))).
```

Y `uyParejaVegetariana (MC Pan (MC Queso (MC Hamburguesa (MC Queso (MC Pan Caja)))))`
debería dar como resultado
`(MC Pan (MC Queso (MC Pan Caja)), MC Pan (MC Hamburguesa (MC Pan Caja))).`

Ejercicio 3. [25 puntos] Implementar `macCapicua:: MacSandwichito -> Bool`, que dado un MacSandwichito, devuelve verdadero si y sólo si éste es capicúa.

```
problema macCapicua (m: MacSandwichito) = result : Bool {
  asegura result == (pisos(m) == reverso(pisos(m)));
}
aux reverso (ls:[T]) : [T] = [ls_{|ls|-1-i}|i ← [0..|ls|)];
```

Ejercicio 4. [10 puntos] (Ejercicio 5.3 tomado de la práctica 7). Dada una lista de elementos de tipo $\mathbb{Z}$ denominada L, definir una función subListasOrdenadas que devuelva:

a) una lista de listas de elementos de tipo $\mathbb{Z}$ tal que cada lista tiene todos sus elementos iguales y longitud igual a la cantidad de apariciones de ese elemento en L, además la lista resultante está "ordenada" ascendentemente. Por ejemplo, $L = [8, 5, 5, 4, 9, 4, 4, 4, 8]$ la función debería devolver [[4,4,4,4],[5,5],[8,8],[9]].