METODOLOGÍA DE LA PROGRAMACIÓN
Escuela de Ingeniería de Bilbao (UPV/EHU)
Grado en Ingeniería Informática de Gestión y Sistemas de Información
28 de abril de 2017
Examen Parcial – Tema 5 – Especificación ecuacional de tipos abstractos de datos
Grupo 01 – 2,5 puntos

EJERCICIO 1 (Especificación ecuacional -- Listas) – (0,750 puntos)

a) (0,550 puntos) **Especificar ecuacionalmente** la función *cc* que dados un entero n, un valor x de tipo t, una lista de tipo t y otro entero p, devuelve la lista que se obtiene colocando n copias del valor x a partir de la posición p. Las posiciones se cuentan empezando desde la izquierda y se considera que el elemento que está más a la izquierda ocupa la posición 1. Si n es negativo, o si p es menor o igual que 0 o p es mayor que la longitud de la lista más 1, entonces se ha de mostrar un mensaje de error. Por tanto, si la lista inicial es vacía, p será adecuado solo si es 1. Para colocar los elementos al final de la lista, el valor de p ha de ser la longitud de la lista más 1.

Además de las operaciones constructoras [] y : se puede utilizar la siguiente función auxiliar sin necesidad de definirla:

- *longitud*, que dada una lista devuelve el número de elementos de la lista.

Ejemplos:

- cc(3, 10, [5, 8, 7], 1) = [10, 10, 10, 5, 8] (se han colocado tres dieces a partir de la posición 1)
- cc(3, 10, [5, 8, 7], 2) = [5, 10, 10, 10, 8, 7] (se han colocado tres dieces a partir de la posición 2)
- cc(3, 10, [5, 8, 7], 3) = [5, 8, 10, 10, 10, 7] (se han colocado tres dieces a partir de la posición 3)
- cc(3, 10, [5, 8, 7], 4) = [5, 8, 7, 10, 10, 10] (se han colocado tres dieces a partir de la posición 4, es decir, al final de la lista)
- cc(3, 10, [], 1) = [10, 10, 10] (se han colocado tres dieces a partir de la posición 1, es decir, al final de la lista)

b) (0,200 puntos) Una vez dadas las ecuaciones, **desarrollar** paso a paso el siguiente ejemplo indicando en cada paso qué ecuación se ha utilizado:

cc(3, 10, [5, 8, 7], 3)

EJERCICIO 2 (Inducción sobre listas) – (1,000 puntos)

a) (0,075 puntos) **Especificar ecuacionalmente** la función "*suma*" que, dada una lista de tipo Int devuelve la suma de los elementos de la lista.

suma :: ([Int]) → Int

Ejemplos:

suma([8, 5, 9, 5]) = 27
suma([]) = 0

b) (0,225 puntos) **Especificar ecuacionalmente** la función "*inversa*" que, dada una lista de tipo t devuelve la lista que se obtiene colocando los elementos en orden inverso.

inversa :: ([t]) → [t]

Ejemplos:
inversa([5, 7, 5, 9]) = [9, 5, 7, 5]
inversa([8, 7, 0]) = [0, 7, 8]
inversa([4]) = [4]
inversa([]) = []

c) (0,100 + 0,600 puntos) **Probar por inducción** que para cualquier lista s de tipo Int se cumple la siguiente propiedad:

suma(s) = suma(inversa(s))

La inducción debe realizarse sobre s considerando el caso básico s **= []** y el caso inductivo s **= z:w**. La hipótesis de inducción consistirá en suponer que para la lista w se cumple la propiedad que se está probando.

Además, hace falta utilizar la propiedad Prop que expresa que para dos listas cualesquiera de enteros u y v se cumple lo siguiente:

(Prop) suma(u ++ v) = suma(u) + suma(v)

EJERCICIO 3 (Especificación ecuacional – Pilas) – (0,200 puntos)

Especificar ecuacionalmente la función *sb* que dados un elemento x de tipo t y una pila de tipo t, sustituye el elemento de la base de la pila por el valor x. Si la pila es vacía, se ha de devolver un mensaje de error.

Además de las funciones constructoras *Pvacia* y *Apilar*, se puede utilizar la siguiente función sin necesidad de definirla:

- *es_pvacia*, que dada una pila devuelve *True* si la pila es vacía y *False* en caso contrario.

Ejemplo 1:

sb(**5** , 7 / 2 / 17 / 9 / 30 / **9**) = 7 / 2 / 17 / 9 / 30 / **5**	*sb*(**2** , 17 / 4 / 30 / **9**) = 17 / 4 / 30 / **2**	*sb*(**6** , **10**) = **6**
El elemento de la base, 9, es sustituido por el valor 5.	**El elemento de la base, 9, es sustituido por el valor 2.**	**El elemento de la base, 10, es sustituido por el valor 6.**

Ejemplo 2:

sb(**5**, Apilar(3, Apilar(**9**, Pvacia))) = Apilar(3, Apilar(**5**, Pvacia))

El elemento de la base, 9, es sustituido por el valor 5.

EJERCICIO 4 (Especificación ecuacional – Colas) – (0,150 puntos)

a) (0,050 puntos) **Especificar ecuacionalmente** la función *estac* que dados un entero x y una cola de enteros, devuelve True si x está en la cola y devuelve False en caso contrario.

Ejemplos:

estac(7, << 5, **7**, 8, 8, **7**, 0 >>) = True	estac(7, << 5, 8, 8, 0 >>) = False
estac(7, Poner(Poner(Cvacia, 2), 9)) = False	estac(7, Cvacia) = False

b) (0,100 puntos) **Especificar ecuacionalmente** la función *cpq* que dada una cola de enteros, devuelve la cola que se obtiene manteniendo solo la copia que esté más a la izquierda para aquellos elementos que estén repetidos. Si la cola inicial es vacía, se ha de devolver la cola vacía.

Ejemplos:

cpq(<< 5, 7, 8, 6, 8, 7, 0 >>) = << 5, 7, 8, 6, 0 >>	cpq(<< 5, 8, 0 >>) = << 5, 8, 0 >>
cpq(Poner(Poner(Poner(Cvacia, 2), 9), 2)) = Poner(Poner(Cvacia, 2), 9)	cpq(Cvacia) = Cvacia

EJERCICIO 5 (Especificación ecuacional – Árboles binarios) – (0,400 puntos)

Especificar ecuacionalmente la función *ch* que dado un árbol binario de tipo t, devuelve el árbol binario que se obtiene colocando una hoja en aquellos nodos que solo tienen un subárbol no vacío, es decir, sustituyendo el árbol vacío por un árbol de un único nodo. La nueva hoja ha de tener el mismo valor que la raíz del otro subárbol. Si el árbol dado como dato de entrada es vacío, la función ha de devolver un árbol vacío.

Además de las operaciones constructoras *Avacio* y *Crear*, se pueden utilizar las función *es_avacio* que decide si un árbol binario es vacío o no, devolviendo True o False según el caso, y la función *raiz* que dado un árbol binario no vacío devuelve el valor de su nodo raíz.

Ejemplo 1:

Figura 1

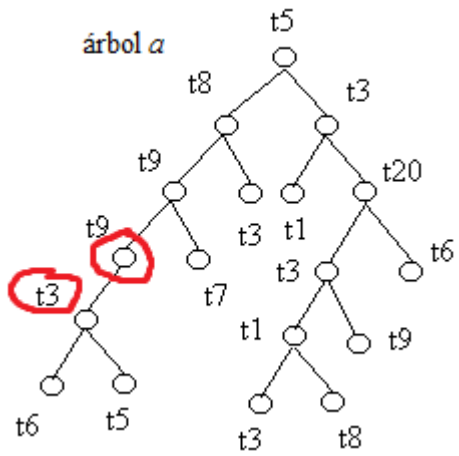


Figura 2

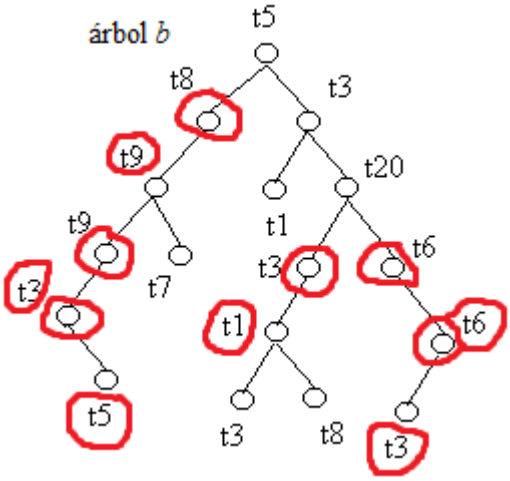


Figura 3

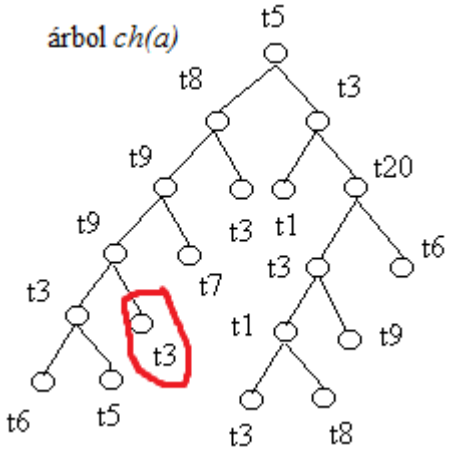
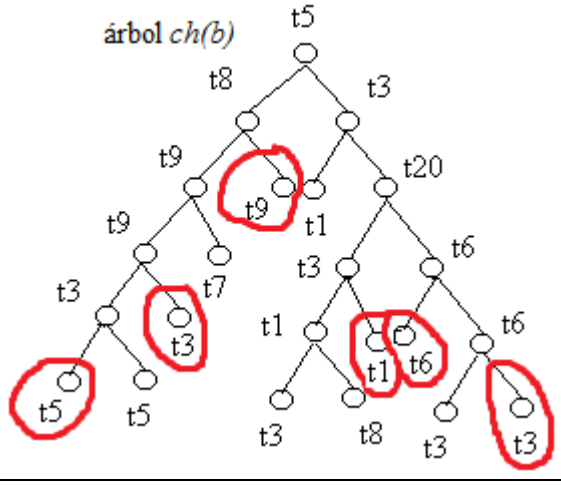


Figura 4



En las Figuras 1 y 2 se muestran los árboles binarios *a* y *b*. En ellos se han marcado los nodos que tienen un único subárbol vacío y los valores que han de tener los nuevos nodos. En las Figuras 3 y 4 se muestran los árboles que se obtendrían al añadir las hojas correspondientes.

Ejemplo 2:

ch(Crear(t8, Crear(t1, Crear(t2, Avacio, Avacio), Avacio), Crear(t3, Avacio, Avacio))
) = Crear(t8, Crear(t1, Crear(t2, Avacio, Avacio), Crear(t2, Avacio, Avacio)),
Crear(t3, Avacio, Avacio))