Facultad de Ciencias Exactas, Ingenier Agrimensura
Instituto Politecnico Superior
Analista Universitario en Sistemas
Estructuras de Datos y Algoritmos II

Nombre y Apellido: Legajo:

1. El inventario de un juego permite que un jugador recolecte diferentes objetos para intercambiarlos dentro del juego. El tad Inv es una colección de tamaño limitado que almacena identificadores ID únicos asociados a una Cantidad que es el número de objetos recolectados de tipo ID. El tamaño de un inventario se determina al crearse este y es la cantidad de IDs que puede almacenar. Este **tad** soporta las siguientes operaciones:

tad Inv (I : Ordered Set)
import List, Bool, Nat, Maybe
new : Nat → Inv I Nat
collect : I → Inv I Nat → Inv I Nat
del : I → Inv I Nat → Inv I Nat
countI : I → Inv I Nat → Maybe Nat
size : Inv I Nat → Nat
fromList : List I → Inv I Nat

donde

- *new* crea un inventario vacío de tamaño n.
- *collect* toma un id e incrementa en uno la cantidad asociada a este, si el id no está en el inventario lo agrega en la siguiente posición vacía si existe una, de lo contrario lo descarta.
- *del* toma un id y decrementa en uno su cantidad asociada a este si el id existe en el inventario.
- *countI* toma un id, un inventario y retorna la *cantidad* asociada al id
- *size* dado un inventario nos dice que tamaño tiene.
- *fromList* dada una lista *ls* que contiene *ids* (posiblemente repetidos), crea un inventario donde entren todos los elementos que están en la lista.
  Ejemplo: sea `['a', 'b', 'a', 'v', 'c', 'b']` la listas de *ids*, el inventario generado por *fromList* debería ser $\{<$`'a'` $2>, <$`'b'` $2>, <$`'v'` $1>, <$`'c'` $1>\}$

Especifique el **tad** inventario, considere definidas las operaciones usuales para los **tad** importados: *cons*, *nil*, *nothing*, *just*, etc. Para el **tad** Nat puede usar las operaciones suma $(+)$ o restas $(-)$ usuales teniendo en cuenta las restricciones que tiene el **tad** Nat

2. Para implementar las operaciones del **tad** Inv se utilizará el tipo:

data Inv a = E | Item (Inv a) (Maybe $(a$, Int$)$) (Inv a)

donde el tamaño del inventario queda definido por:

size :: Inv $a \to$ Int
size E $= 0$
size (Item l e r) $= 1 +$ *size* $l +$ *size* r

, el tipo Inv está **balanceado** y los elementos se completan según el recorrido *inorder* del inventario. Implementar las operaciones *new* y *collect* del **tad** Inv en haskell.

3. Dadas las siguientes definiciones:

atFirst x *xss* $= [\,] :$ *map* $(x:)$ *xss*
atLast $[ys]$ $x = [ys, ys +\!\!+ [x]]$
atLast $ys : yss$ $x = ys :$ *atLast* *yss* x

a) Determinar los tipos de las funciones *atFirst* y *atLast*.

b) Demostrar que para toda lista *yss* y para todos elementos x e x', se cumple que:

$$atFirst\ x\ (atLast\ yss\ x') = atLast\ (atFirst\ x\ yss)\ x'$$

4. Dada la siguiente recurrencia:

$$W_f(1) = 1$$
$$W_f(n) = 3W_f(\frac{n}{3}) + n^3$$

a) Resolver la recurrencia expandiendo la definición en forma algebraica.

b) Utilizando lo resuelto en el ejercicio anterior dibujar el árbol de recurrencia con los valores por niveles y realizar las operaciones necesarias para alcanzar los mismos valores.

c) De ser posible confirma el resultado del primer ítem utilizando el teorema maestro o por el contrario justificar por que no es posible usarlo.