Algoritmos y Estructuras de Datos II
Recuperatorio del primer parcial – 29 de noviembre de 2014

Aclaraciones

- El parcial es **a libro abierto**.
- Cada ejercicio debe entregarse **en hojas separadas**.
- Incluir en cada hoja el número de orden asignado, número de hoja, apellido y nombre.
- Al entregar el parcial, completar el resto de las columnas en la planilla.
- Cada ejercicio se calificará con **MB**, **B**, **R** o **M**, y podrá recuperarse independientemente de los demás. Para aprobar el parcial se puede tener hasta 1 (un) ejercicio **R** (regular), siempre que no se trate del ejercicio 1. Para más detalles, ver "Información sobre la cursada" en el sitio Web.

Ej. 1. Especificación

Se quiere especificar un sistema para controlar una oficina postal que se ocupa de almacenar, administrar y entregar la correspondencia en una ciudad. La ciudad en cuestión se encuentra dividida en zonas, cada una de las cuales se identifica con un código postal. La oficina recibe periódicamente un cargamento de paquetes (cartas, encomiendas, documentos, etc.). Cada paquete tiene escrito el código postal de su destinatario y un peso en gramos.

La oficina cuenta con un grupo de empleados destinados al reparto de paquetes (los *carteros*). A cada cartero le corresponde entregar los paquetes dentro de una única zona. Es decir, cada cartero tiene asociado un único código postal.

Cuando llega un camión con un cargamento, los empleados de la oficina organizan los paquetes recibidos. Los paquetes cuyo código postal no se corresponde con el de algún cartero se devuelven inmediatamente al camión. Los paquetes restantes ingresan al depósito del correo.

Un cartero puede iniciar su recorrido en cualquier momento, llevándose del depósito una bolsa de paquetes para repartir. Los paquetes a repartir se asignan de la manera detallada a continuación, teniendo en cuenta que el método utilizado para seleccionarlos se postergará hasta el momento de la implementación:

- A cada cartero le corresponden únicamente paquetes asociados a su código postal.
- El peso total de la bolsa no debe exceder los 25Kg.

Cuando el cartero finaliza su recorrido, reporta cuáles paquetes no pudieron ser entregados. Los paquetes *rebotados* se reingresan al depósito. Este conjunto debe ser (obviamente) un subconjunto de los paquetes que le correspondía entregar al momento de iniciar su recorrido, en caso de que sea el mismo conjunto, el cartero es **despedido** de la oficina postal.

Especificar el sistema utilizando TADs. Se desea saber en todo momento:

- Cuántos paquetes hay en el depósito, y qué características tienen.
- Cuántos paquetes fueron rebotados.

Ej. 2. Inducción estructural

En cierta ciudad robótica se cuenta con el siguiente tipo abstracto de datos:

TAD RESTRICCIÓN

géneros restr

generadores

$\langle\bullet\rangle$	:	tag	$\longrightarrow$	restr
NOT $\bullet$	:	restr	$\longrightarrow$	restr
$\bullet$ **AND** $\bullet$	:	restr $\times$ restr	$\longrightarrow$	restr
$\bullet$ **OR** $\bullet$	:	restr $\times$ restr	$\longrightarrow$	restr

observadores básicos

v?	:	restr $\times$ conj(tag)	$\longrightarrow$	bool

otras operaciones

neg	:	restr	$\longrightarrow$	restr

axiomas $\forall r, r_1, r_2$: restr. $\forall t$: tag. $\forall ts$: conj(tag).

Vtag)	v?($\langle t\rangle$, ts)	$\equiv$	$t \in ts$
Vnot)	v?(**NOT** r, ts)	$\equiv$	$\neg$v?(r, ts)
Vand)	v?(r_1 **AND** r_2, ts)	$\equiv$	v?(r_1, ts) $\wedge$ v?(r_2, ts)
Vor)	v?(r_1 **OR** r_2, ts)	$\equiv$	v?(r_1, ts) $\vee$ v?(r_2, ts)
Ntag)	neg($\langle t\rangle$)	$\equiv$	**NOT** $\langle t\rangle$
Nnot)	neg(**NOT** r)	$\equiv$	r
Nand)	neg(r_1 **AND** r_2)	$\equiv$	neg(r_1) **OR** neg(r_2)
Nor)	neg(r_1 **OR** r_2)	$\equiv$	neg(r_1) **AND** neg(r_2)

Fin TAD

Se quiere demostrar por inducción estructural la siguiente propiedad:

$$(\forall r : \text{restr})(\forall ts : \text{conj(tag)})(\text{v?}(\text{neg}(\text{neg}(r)), ts) = \text{v?}(r, ts))$$

a) Escribir el predicado unario. Luego escribir, completo, **el esquema de inducción** a utilizar.
En el esquema, marcar **claramente** CB(s), PI(s), HI(s), TI(s) y el alcance de cada cuantificador.

b) Plantear el/los caso(s) base y resolverlo(s), justificando cada paso de la demostración.

c) Plantear el/los paso(s) inductivo(s) y resolverlo(s), justificando cada paso de la demostración. **Nota:** se pueden usar sin demostrar propiedades lógicas que no involucren restricciones. Aclarar explícitamente en qué pasos de la demostración se recurre a ellas. **Nota II:** en caso de ser necesario recurrir a un lema auxiliar para demostrar el paso inductivo, enunciarlo claramente y demostrarlo por inducción estructural.

Ej. 3. Rep y Abs

En este ejercicio se quiere diseñar un diccionario que está optimizado para responder con mayor eficiencia las consultas más recientes. Para ello consideraremos el siguiente tipo abstracto de datos, que representa un diccionario que mantiene un registro de cuál fue la última vez que se accedió a una clave:

TAD DiccionarioConHistorialAccesos(clave, valor)

géneros dha

generadores

nuevo	:		⟶ dha	
definir	:	clave × valor × dha	⟶ dha	
acceder	:	clave × dha	⟶ dha	{def?(c, d)}

observadores básicos

def?	:	clave × dha	⟶ bool	
obtener	:	clave c × dha d	⟶ valor	{def?(c,d)}
#accesos	:	dha d	⟶ nat	
últAcceso	:	clave c × dha d	⟶ nat	{def?(c,d)}

axiomas

∀ d : dha. ∀ c, c′ : clave. ∀ v : valor.

def?(c, nuevo)	≡	false
def?(c, definir(c′, v, d))	≡	c = c′ ∨_L def?(c, d)
def?(c, acceder(c′, d))	≡	def?(c, d)

Fin TAD

obtener(c, definir(c′, v, d))	≡	**if** c = c′ **then** v **else** obtener(c, d) **fi**
obtener(c, acceder(c′, d))	≡	obtener(c, d)
#accesos(nuevo)	≡	0
#accesos(definir(c, v, d))	≡	1 + #accesos(d)
#accesos(acceder(c, d))	≡	1 + #accesos(d)
últAcceso(c, definir(c′, v, d))	≡	**if** c = c′ **then** #accesos(d) + 1 **else** últAcceso(c, d) **fi**
últAcceso(c, acceder(c′, d))	≡	**if** c = c′ **then** #accesos(d) + 1 **else** últAcceso(c, d) **fi**

En el contexto del módulo que se quiere diseñar, se accede a una clave en cualquier situación que requiera ubicarla en la estructura interna, ya sea mediante la operación Definir que asocia una clave a un valor, o mediante las operaciones Def? y Obtener que consultan el valor asociado a una clave.

El módulo DHA se representa con *estr*, donde:

$$estr \text{ es tupla} \left(\begin{array}{l} \#accesos\text{: nat}, \\ completo\text{: diccionario(clave, info)}, \\ capacidad\text{: nat}, \\ caché\text{: diccionario(clave, info)}, \\ por\acute{U}ltAcceso\text{: diccionario(nat, clave)} \end{array} \right) \qquad info \text{ es tupla} \left(\begin{array}{l} val\text{: valor} \\ \acute{u}ltAcceso\text{: nat} \end{array} \right)$$

La idea de la estructura es la siguiente: *#accesos* indica cuántos accesos se realizaron en total, desde la creación del DHA. El campo *completo* es un diccionario que almacena todas las entradas definidas en el DHA. El diccionario *caché* es otro diccionario que almacena las k entradas accedidas más recientemente, donde k es el número indicado por el campo *capacidad* de la estructura. En caso de que haya menos de k entradas en el DHA, el diccionario *caché* las contiene a todas. El diccionario *por ÚltAcceso* indica, para cada una de las claves de la *caché*, el momento en que fue accedida por última vez. Tener en cuenta que el diccionario *completo* puede encontrarse **desactualizado** si alguna clave se encuentra en la *caché* y se modificó su valor.

a) Escribir el invariante de representación en castellano.

b,c) Escribir formalmente *b)* el invariante de representación y *c)* la función de abstracción.

Nota: para darse una idea de cómo se mantiene la estructura cuando se quiere definir, consultar, u obtener una clave:

- Se busca la clave en la *caché*.
- En caso de no estar presente allí, se la busca en el diccionario *completo*.
- Si la clave no estaba en la *caché*, se la inserta allí, porque se trata de una de las claves accedidas recientemente.
- Después, se incrementa el contador *#accesos* para indicar que hubo un acceso más.
- Luego, se modifica el valor *últAcceso* asociado a la clave, para indicar que fue consultada en el momento actual (el número indicado por *#accesos*), y se actualiza el diccionario *por ÚltAcceso*.
- Finalmente, si la *caché* está llena (tiene más de k entradas), se elimina la clave más antigua de la *caché* y de *por ÚltAcceso*, y se actualiza su información en el diccionario *completo*.