Algoritmos y Estructuras de Datos II
Primer parcial – Sábado 23 de Abril de 2016

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 **P**romocionado, **A**probado, **R**egular, o **I**nsuficiente.
- El parcial completo está aprobado si el primer ejercicio tiene al menos **A**, y entre los ejercicios 2 y 3 hay al menos una **A**. Para más detalles, ver "Información sobre la cursada" en el sitio Web.

Ej. 1. Especificación

El **buscaminas** es un juego cuyo objetivo es encontrar las posiciones con minas en una grilla rectangular. Llamaremos a esta *tablero*, mientras que llamaremos a sus posiciones *celdas*. Hay tres tipos de celdas: las *minadas*, que son aquellas que tienen una mina, las *pistas*, que son aquellas que son *lindantes* (en sentido horizontal, vertical o diagonal) a al menos una mina, y las *libres*, que son aquellas que no son ni minadas ni pistas. La Figura 1 muestra un ejemplo de un tablero de buscaminas.

			1	⋆
1	1		2	2
⋆	1		1	⋆
1	1		1	1
	1	1	1	
	1	⋆	1	

Figura 1: Tablero inicial, con todas sus celdas tapadas (grises). [⋆]: mina; [1]: valor de una celda pista; las celdas vacías son celdas libres.

La mecánica del juego es la siguiente. Inicialmente, se dispone una cierta cantidad de minas sobre distintas posiciones del tablero. Éste se le ofrece al jugador, con todas las celdas *tapadas*[1]. Luego, iterativamente, el jugador elige *revelar* una a una las distintas celdas. El juego termina cuando o bien: a) el jugador logra revelar todas las celdas que no son minas (ganando el juego), o b) el jugador revela alguna celda minada (perdiendo).

Cuando el jugador decide revelar una celda (x,y), si ésta no está libre, entonces *únicamente* se revela (x,y). En cambio, si (x,y) es una celda libre, no sólo hay que revelar (x,y), sino tambien todas sus celdas lindantes. Notar que este es un proceso recursivo; si alguna celda (x', y') lindante a (x,y) tambien está libre, entonces se revela dicha celda también[2]. Vale notar que **no** hay posibilidades de que se revele una celda minada en este proceso, aunque sí se revelarán todas las celdas pista que se encuentran al "borde" de la región formada por celdas libres (ver Figura 2).

Las *celdas pista* agregan estrategia al juego. Cada una está asociada a un número que llamaremos *la pista*, que se define como la cantidad de celdas minadas que son lindantes a la pista. Luego, a medida que el jugador va liberando las distintas pistas, puede usarlas para determinar qué posiciones tienen minas a fin de evitarlas.

Se pide modelar con un TAD el juego del buscaminas. En particular, se desea saber: **qué celdas están reveladas**; **si el juego terminó** y, en caso afirmativo, **con qué resultado**; y la pista asociada a cualquier celda pista ya revelada.

Además, se debe proveer una función llamada **seguroLindaAMinas** que, dado un buscaminas y la posición de una celda pista c ya revelada, indique si la cantidad de celdas lindantes a la pista c que no están reveladas, coincide con el valor de la pista. En el ejemplo de la Figura 2, la función deber devolver *true* para celda inferior a la destapada, y *false* para la inferior a esta última.

	?		1	⋆
1	1		2	2
⋆	1		1	⋆
1	1		1	1
	1	1	1	
	1	⋆	1	

Figura 2: Tablero luego de destapar la celda [?], resaltando en blanco todas las celdas destapadas por esa acción.

Ej. 2. Inducción estructural

Dadas las siguientes definiciones:

derechista : ab(α) $\longrightarrow$ bool

d_0) derechista(a) $\equiv$ nil?(a) $\vee_L$ (nil?(izq(a)) $\wedge$ derechista(der(a)))

inorder : ab(α) $\longrightarrow$ secu(α)

IO) inorder(a) $\equiv$ **if** nil?(a) **then** <> **else** inorder(i) & (e • inorder(d)) **fi**

postorder : ab(α) $\longrightarrow$ secu(α)

PO) postorder(a) $\equiv$ **if** nil?(a) **then** <> **else** (postorder(i) & postorder(d))∘ e **fi**

Demuestre por inducción estructural la siguiente propiedad sobre árboles binarios:

$$\boxed{(\forall a : \text{ab}(\alpha))\ ((\text{derechista}(a) \Rightarrow \text{inorder}(a) \equiv \text{reverso}(\text{postorder}(a)))}$$

[1] En otras palabras, el jugador inicialmente no conoce el tipo de ninguna celda

[2] **Hint:** al buscar los casilleros a revelar, lleve el registro de qué casilleros ya fueron visitados para no repetirse.

En caso de utilizar lemas auxiliares, plantearlos claramente y demostrarlos. Además, se pide:

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.

Ej. 3. Diseño

Egresados de esta facultad quieren implementar una versión local de uno de los juegos de acción más populares de los últimos tiempos. El siguiente TAD modela el comportamiento de la versión *multiplayer online* del mismo.

TAD Jug es string, **TAD** Nivel es Nat, **TAD** Puntaje es Nat

TAD Carmen Ordena Delivery

géneros cod

observadores básicos

jugadores	: cod	$\longrightarrow$	conj(jug)	
defNiveles	: cod	$\longrightarrow$	secu(puntaje)	
puntaje	: cod c × jug j	$\longrightarrow$	puntaje	{j ∈ jugadores(c)}

generadores

sistemaOnline	: conj(jug) js × secu(puntaje) $defNivel$	$\longrightarrow$	cod	{long(defNivel) ¿0 ∧ ordenadaEstricta(defNivel)}
peleaDelivery	: cod c × dicc(jug, dicc(jug, nat)) $palos$	$\longrightarrow$	cod	$\left\{\begin{array}{l}\text{claves(palos)} \subseteq \text{jugadores(c)} \wedge ((\forall j\text{: jug})\ \text{def?(j,palos)} \Rightarrow_L \text{claves(obtener}(j, palos)) =_{obs} \text{claves(palos)} \wedge \\ \text{obtener(j, obtener}(j, palos)) = 0)\end{array}\right\}$

otras operaciones

nivel	: cod c × jug j	$\longrightarrow$	nivel	{j ∈ jugadores(c)}
valores	: dicc(α, β)	$\longrightarrow$	secu(β)	
ordenadaEsctricta	: secu(nat)	$\longrightarrow$	bool	

axiomas

jugadores(sistemaOnline(js, n))	≡	js
jugadores(peleaDelivery($c, palos$))	≡	jugadores(c)
defNiveles(sistemaOnline(js, n))	≡	n
defNiveles(peleaDelivery($c, palos$))	≡	defNiveles(c)
puntaje(sistemaOnline($js, defNivel$),j)	≡	0
puntaje(peleaDelivery($c, palos$),j)	≡	puntaje(c, j) + sumar(valores(obtener($j, palos$)))
sumar(s)	≡	**if** vacia?(s) **then** 0 **else** prim(s) + sumar(fin(s)) **fi**
valores(d)	≡	**if** ∅?(claves(d)) **then** <> **else** obtener(dameUno(claves(d)),d) • valores(borrar(dameUno(claves(d)),d)) **fi**
ordenadaEstricta(s)	≡	**if** long(s) < 2 **then** *true* **else** prim(s) < prim(fin(s)) ∧ ordenadaEstricta(fin(s)) **fi**

Fin TAD

El TAD aquí presentado modela el sistema *online* para el juego *Carmen Ordena Delivery*, que trata de una señora que pide mucho delívery y da buena propina. Los jugadores interpretan a chicos y chicas de delivery en bicicleta. Cada pelea representa una vez que Carmen pide delivery. En esa pelea participan un subconjunto de jugadores. Los jugadores pueden ponerse palos en las ruedas. Cada vez que un jugador pone palos en la rueda a otros, el primero suma su puntaje. En el sistema online además hay un sistema de niveles. Cada jugador posee un nivel. El nivel al que pertenece cada jugador está definido por su puntaje. La secuencia $defNivel$ que recibe el TAD en el generador base define en $defNivel[i]$ el puntaje mínimo para alcanzar el nivel i. Esta secuencia debe estar ordenada de forma creciente y estricta para que siempre esté definido a qué nivel pertenece un jugador.

Para representar el TAD Carmen Ordena Delivery se decidió utilizar la siguiente estructura: cod **se representa con** estr, donde

estr es tupla ⟨ *jugadores*: conj(jug),
puntajeANivel: dicc(puntaje, nivel),
nivelJugador: dicc(jug, nivel),
palosTot: dicc(jug, dicc(jug, nat))

donde *jugadores* es el conjunto de jugadores del sistema online, *puntajeANivel* nos dice para cada puntaje que sube el nivel del jugador a qué nivel sube el mismo, *nivelJugador* es el nivel para cada jugador y *palosTot* es un diccionario donde para cada jugador tiene un diccionario diciendo para cada otro jugador, cuantos palos le puso en las ruedas en total.

a) Escribir en castellano el invariante de representación.

b) Escribir formalmente el invariante de representación.

c) Escribir formalmente la función de abstracción.