Algoritmos y Estructuras de Datos II
Recuperatorio del primer parcial – 30 de junio de 2017

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 estará aprobado si: (el ejercicio 1 tiene al menos **A**) y (el ejercicio 3 tiene al menos una **A** o (el ejercicio 3 tiene una **R** y el 2 al menos una **A**)).

Ej. 1. Especificación

Un sistema operativo (SO) ejecuta muchos programas en simultáneo y mantiene un conjunto de recursos dedicados que los programas utilizan. En cualquier momento pueden iniciarse o finalizarse programas. Además, en cualquier momento un programa puede solicitar el uso de uno de los recursos del sistema. Si el recurso está libre, el SO se lo asigna en forma dedicada y el programa continúa. Si el recurso estaba asignado a algún otro proceso, el SO bloquea al proceso que solicita el recurso y lo deja a la espera de que tal recurso se libere. El proceso bloqueado no podrá hacer nada (ni siquiera terminar su ejecución) hasta que el SO le asigne el recurso solicitado. Cuando un proceso termina de utilizar un cierto recurso, lo informa al SO y éste asigna automáticamente el recurso a algún otro proceso que esté bloqueado esperando por tal recurso (de haberlo). Si hubiese más de un proceso bloqueado esperando por tal recurso, el SO elige cualquiera de ellos. Un proceso solo puede finalizar si ya liberó todos los recursos que estaba usando.

Dada una secuencia de recursos $r_1, \ldots, r_k$ y una de procesos $p_1, \ldots, p_k$, podría suceder que p_1 esté utilizando el recurso r_1 y que para todo $i = 1, \ldots, k-1$, p_i esté esperando el recurso r_{i+1} el cual está siendo utilizado por p_{i+1}. La Figura 1 muestra un ejemplo de esta situación con $k = 6$, en la cual una flecha de un recurso r a un proceso p indica que r está siendo utilizado por p y la flecha inversa indica que p está esperando que se libere r. En principio no hay ningún problema con esta situación, pero si en tal caso el proceso p_k solicitara el recurso r_1, esto formaría un ciclo de solicitudes en el cual todos los procesos involucrados estarían bloqueados; esta condición se conoce como *deadlock*. El SO especificado debe ofrecer la funcionalidad de saber si una solicitud de un proceso p por un recurso r generaría un deadlock o no, dado el estado actual del SO. Además, el SO debería restringir sus operaciones para que esto no suceda nunca.

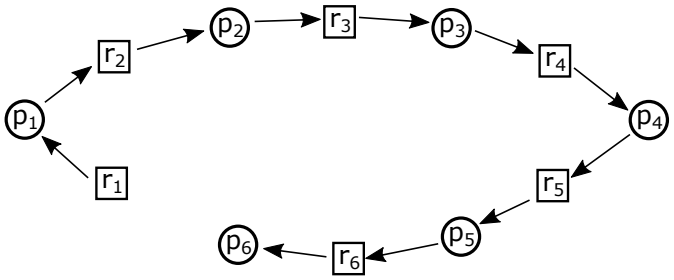

Figura 1: Ejemplo en el cual si p_6 solicita r_1 se produce un deadlock.

Se pide modelar con un TAD el Sistema Operativo descripto, teniendo en cuenta, entre otras cosas, la funcionalidad de saber si una solicitud de un proceso p por un recurso r generaría un deadlock o no (y restringiendo adecuadamente las operaciones para que esto no ocurra nunca).

Ej. 2. Complejidad

Discutir la veracidad de las siguientes afirmaciones, justificando adecuadamente en cada caso:

1. Si $f(n) = n + \log^2 n$ y $g(n) = n + \log n$ entonces $f(n) \in O(g(n))$.
2. Si $f(n) = n2^n$ y $g(n) = 3^n$ entonces $f(n) \in \Omega(g(n))$.
3. Si $f(n) = 2^{2n}$ entonces $f(n) \in O(2^n)$.
4. Si $f(n) = \frac{n}{\log n}$ y $g(n) = (\log n)^{\log n}$ entonces $f(n) \in O(g(n))$.

Ej. 3. Diseño

El grupo FUMBOL se dedica a organizar torneos de fútbol entre equipos amateurs. Los equipos se registran al iniciar el torneo informando los jugadores de su plantel. A lo largo del torneo los equipos se enfrentan entre sí, aunque cada par de equipos no se enfrenta más de una vez. De cada partido jugado se registran los goles de cada equipo (registrando qué jugadores hicieron los mismos). El siguiente TAD (del cual se omiten las axiomatizaciones) modela el torneo descripto.

TADs EQUIPO y JUGADOR son STRING

TAD FUMBOL

observadores básicos

equipos	:	fumbol	$\longrightarrow$	conj(equipo)	
plantel	:	fumbol f × equipo e	$\longrightarrow$	conj(jugador)	$\{ e \in \text{equipos}(f)\}$
jugóContra	:	fumbol f × equipo e_1 × equipo e_2	$\longrightarrow$	bool	$\{ \{e_1, e_2\} \subseteq \text{equipos}(f) \}$
golesContra	:	fumbol f × equipo e_1 × equipo e_2	$\longrightarrow$	secu(jugador)	$\{\{e_1, e_2\} \subseteq \text{equipos}(f) \wedge_{\text{L}} \text{jugóContra}(f, e_1, e_2)\}$

generadores

iniciarTorneo	:	dicc(equipo, conj(jugador)) d	$\longrightarrow$	fumbol	{plantelesDisjuntos(d)}
resultado	:	fumbol f × equipo e_1 × equipo e_2 × secu(jugador) g_1 × secu(jugador) g_2	$\longrightarrow$	fumbol	$\{e_1 \neq e_2 \wedge \{e_1, e_2\} \subseteq \text{equipos}(f) \wedge_{\text{L}} \neg\text{jugóContra}(f, e_1, e_2) \wedge g_1 \subseteq \text{plantel}(f, e_1) \wedge g_2 \subseteq \text{plantel}(f, e_2)\}$

otras operaciones

goleadores : fumbol $\longrightarrow$ conj(jugador)

axiomas

...

Fin TAD

Se decidió utilizar la siguiente estructura para representar el TAD:

FUMBOL **se representa con** `estr`

donde `estr` es tupla $\langle$ *equipos*: conj(equipo),
planteles: dicc(equipo, conj(jugador)),
partidos: dicc(equipo, conj(partido)),
goles: dicc(jugador, nat)

y `partido` es tupla$\langle$*rival*: equipo, *golesPropios*: secu(jugador), *golesRival*: secu(jugador)$\rangle$

En esta estructura, *equipos* almacena los equipos del torneo y *planteles* los jugadores de cada equipo. Por otro lado, *partidos* guarda para cada equipo la información de todos los partidos que disputó (la cual incluye, para cada partido, el equipo rival y las listas de goles propios y del rival). Finalmente, se registra en *goles* la cantidad total de goles de cada jugador.

Teniendo en cuenta el TAD FUMBOL y la estructura elegida para su representación se pide:

a) Escribir en castellano el invariante de representación.

b) Escribir formalmente el invariante de representación.

c) Escribir formalmente la función de abstracción.