PROGRAMACION CONCURRENTE - EXAMEN 15-9-2010

En todos los casos, responda con claridad y sintéticamente. En los casos que corresponda, NO SE CONSIDERARAN RESPUESTAN SIN JUSTIFICAR. Tiempo Máximo 2 hs 15 min.

1. Describa los siguientes paradigmas de resolución de programas concurrentes: paralelismo iterativo, paralelismo recursivo, productores y consumidores, clientes y servidores, y peers. Ejemplifique en cada caso.

2. a) Qué significa el problema de "interferencia" en programación concurrente? Cómo puede evitarse?
b) En qué consiste la propiedad de "A lo sumo una vez" y qué efecto tiene sobre las sentencias de un programa concurrente? De ejemplos de sentencias que cumplan y de sentencias que no cumplan con ASV.

3. En los protocolos de acceso a sección crítica vistos en clase, cada proceso ejecuta el mismo algoritmo. Una manera alternativa de resolver el problema es usando un proceso coordinador. En este caso, cuando cada proceso SC[i] quiere entrar a su sección crítica le avisa al coordinador, y espera a que éste le de permiso. Al terminar de ejecutar su sección crítica, el proceso SC[i] le avisa al coordinador.
   Desarrolle protocolos para los procesos SC[i] y el coordinador usando sólo variables compartidas (no tenga en cuenta la propiedad de eventual entrada).

4. a) Defina el concepto de "sincronización barrier". Cuál es su utilidad?
b) Qué es una barrera simétrica?
c) Describa "combining tree barrier" y "butterfly barrier". Marque ventajas y desventajas en cada caso.

5. Dados los siguientes dos segmentos de código, indicar para cada uno de los ítems si son equivalentes o no. Justificar cada caso (de ser necesario dar ejemplos).

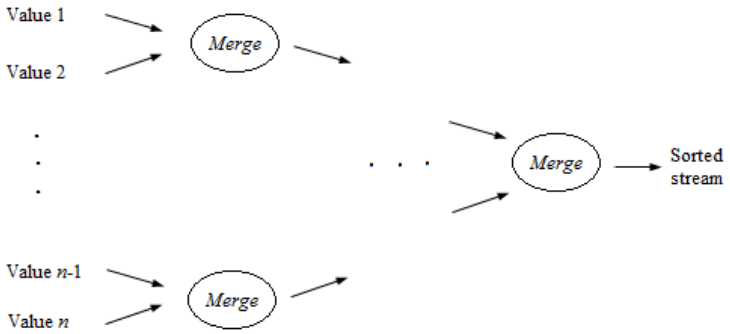
Segmento 1	Segmento 2

...
int cant=1000;
DO (cant < -10); datos?(cant) ⟶
Sentencias1
□ (cant > 10); datos?(cant) ⟶
Sentencias2
□ (INCOGNITA); datos?(cant) ⟶
Sentencias3
END DO
...
	...
	int cant=1000;
	While (true)
	{ IF (cant < -10); datos?(cant) ⟶
	Sentencias1
	□ (cant > 10); datos?(cant) ⟶
	Sentencias2
	□ (INCOGNITA); datos?(cant) ⟶
	Sentencias3
	END IF
	}
	...

   a) *INCOGNITA* equivale a: *(cant = 0),*
   b) *INCOGNITA* equivale a: *(cant > -100)*
   c) *INCOGNITA* equivale a: *((cant > 0) or (cant < 0))*
   d) *INCOGNITA* equivale a: *((cant > -10) or (cant < 10))*
   e) *INCOGNITA* equivale a: *((cant >= -10) or (cant <= 10))*

6. Resuelva con monitores el siguiente problema: Tres clases de procesos comparten el acceso a una lista enlazada: searchers, inserters y deleters. Los searchers sólo examinan la lista, y por lo tanto pueden ejecutar concurrentemente unos con otros. Los inserters agregan nuevos items al final de la lista; las inserciones deben ser mutuamente exclusivas para evitar insertar dos items casi al mismo tiempo. Sin embargo un insert puede hacerse en paralelo con uno o más searchers. Por último, los deleters remueven items de cualquier lugar de la lista. A lo sumo un deleter puede acceder la lista a la vez, y el borrado también debe ser mutuamente exclusivo con searchers e inserciones.

7. Suponga los siguientes métodos de ordenación de menor a mayor para n valores (n par y potencia de 2), utilizando pasaje de mensajes:



i- Un pipeline de filtros. El primero hace input de los valores de a uno por vez, mantiene el mínimo y le pasa los otros al siguiente. Cada filtro hace lo mismo: recibe un stream de valores desde el predecesor, mantiene el más chico y pasa los otros al sucesor.

ii- Una red de procesos filtro (como la de la figura).

iii- Odd/even Exchange sort. Hay n procesos P[1:n], Cada uno ejecuta una serie de rondas. En las rondas "impares", los procesos con número impar P[impar] intercambian valores con P[impar+1]. En las rondas "pares", los procesos con número par P[par] intercambian valores con P[par+1] (P[1] y P[n] no hacen nada en las rondas "pares"). En cada caso, si los números están desordenados actualizan su valor con el recibido.

Nota: Cada proceso tiene almacenamiento local sólo para 2 valores (el próximo y el mantenido hasta ese momento).

a) Cuántos procesos son necesarios en i e ii? Justifique.

b) Cuántos mensajes envía cada algoritmo para ordenar los valores? Justifique.

c) En cada caso, cuáles mensajes pueden ser enviados en paralelo (asumiendo que existe el hardware apropiado) y cuáles son enviados secuencialmente? Justifique.

d) Cuál es el tiempo total de ejecución de cada algoritmo? Asuma que cada operación de comparación o de envío de mensaje toma una unidad de tiempo. Justifique.