PROGRAMACION CONCURRENTE - EXAMEN FINAL - 22-4-2009

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. Sea "cantidad" una variable entera inicializada en 0 que representa la cantidad de elementos de un buffer, y sean P1 y P2 dos programas que se ejecutan de manera concurrente, donde cada una de las instrucciones que los componen son atómicas.

P1::	P2::

if (cantidad = 0) then
begin
cantidad:= cantidad + 1;
buffer := elemento_a_agregar;
end;
	if (cantidad > 0) then
	begin
	elemento_a_sacar:= buffer;
	cantidad:= cantidad - 1;
	end;

Además existen dos alumnos de concurrente que analizan el programa y opinan lo siguiente:
"Pepe: este programa funciona correctamente ya que las instrucciones son atómicas".
"José: no Pepe estás equivocado, hay por lo menos una secuencia de ejecución en la cual funciona erróneamente"
Con cuál de los dos alumnos está de acuerdo? Si está de acuerdo con Pepe justifique su respuesta. Si está de acuerdo con José encuentre una secuencia de ejecución que verifique lo que José opina y escríbala, y modifique la solución para que funcione correctamente (Suponga buffer y elemento variables declaradas).

2. a) A qué se denomina propiedad de programa? Qué son las propiedades de vida y seguridad? Ejemplifique.
b) Defina fairness. Relacione con las políticas de scheduling (NO DESCRIBA LOS DISTINTOS TIPOS DE FAIRNESS).
c) Describa las propiedades que debe cumplir un protocolo de E/S a una sección crítica?
d) Cuáles son los defectos que presenta la sincronización por busy waiting? Diferencie esta situación respecto de los semáforos.
e) Explique la semántica de la instrucción de grano grueso AWAIT y su relación con instrucciones tipo Test & Set o Fetch&Add.

3. Sea la siguiente solución propuesta al problema de alocación LJN (Longest Job Next):

```
monitor LJN {
   bool libre = true;
   cond turno;
   procedure request(int tiempo) {
      if (not libre) wait(turno, (MAXVALOR - tiempo));
      libre = false;       }
   procedure release() {
      libre = true;
      signal(turno);       }
}
```

a) Funciona correctamente con disciplina de señalización Signal and Continue? Justifique.
b) Funciona correctamente con disciplina de señalización Signal and Wait? Justifique

Nota: suponga que MAXVALOR es el máximo valor de tiempo por el cual un proceso puede solicitar el recurso.

4. a) En qué consiste la técnica de "passing the baton" y cuál es su utilidad? Qué relación tiene con "passing the condition"?
b) Utilice la técnica de "passing the condition" para implementar un semáforo fair usando monitores.

5. En qué consiste la comunicación guardada (introducida por CSP) y cuál es su utilidad? Describa cómo es la ejecución de sentencias de alternativa e iteración que contienen comunicaciones guardadas.

6. a) Describa brevemente en qué consisten los mecanismos de RPC y Rendezvous. Para qué tipo de problemas son más adecuados?
b) Por qué es necesario proveer sincronización dentro de los módulos en RPC? Cómo puede realizarse esta sincronización?
c) Qué elementos de la forma general de rendezvous no se encuentran en el lenguaje ADA?

7. Explique sintéticamente los 7 paradigmas de interacción entre procesos en programación distribuida. Ejemplifique en cada caso.

8. a) Cuál es el objetivo de la programación paralela?
b) Mencione las tres técnicas fundamentales de la computación científica. Ejemplifique.
c) Defina las métricas de speedup y eficiencia. Cuál es el significado de cada una de ellas (que miden)? Ejemplifique.
d) En qué consiste la "ley de Amadhi"?

9. 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.