PROGRAMACION CONCURRENTE - EXAMEN FINAL 14-10-2009

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. Dado el siguiente programa concurrente con memoria compartida:

```
x = 2; y = 3; z = 4;
co
  x = x - 2 //  y = z * x   // z = z + y
oc
```

a) En qué consiste la propiedad de "A lo sumo una vez" (ASV)?

b) Cuáles de las asignaciones dentro de la sentencia co cumplen la propiedad de ASV. Justifique claramente.

c) Indique los resultados posibles de la ejecución. Justifique.

Nota 1: las instrucciones NO SON atómicas.

Nota 2: no es necesario que liste TODOS los resultados.

2.

a) A qué se denomina propiedad de programa? Qué son las propiedades de vida y seguridad? Ejemplifique.

b) Defina fairness. Relacione dicho concepto con las políticas de scheduling.

c) Cuáles son las propiedades que debe cumplir un protocolo de E/S a una sección crítica? Cuáles son de seguridad y cuáles de vida?

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. a) Qué significa que un problema sea de "exclusión mutua selectiva"?

b) El problema de los lectores-escritores es de exclusión mutua selectiva? Por qué?

c) De los problemas de los baños planteados en teoría, cuál podría ser de exclusión mutua selectiva? Por qué?

d) Por qué el problema de los filósofos es de exclusión mutua selectiva? Si en lugar de 5 filósofos fueran 3, el problema seguiría siendo de exclusión mutua selectiva? Por qué?

e) El problema de los filósofos resuelto en forma centralizada y sin posiciones fijas es de exclusión mutua selectiva? Por qué?

f) Si en el problema de los lectores-escritores se acepta sólo 1 escritor o 1 lector en la BD, tenemos un problema de exclusión mutua selectiva? Por qué?

4.

Una manera de ordenar *n* enteros es usar el algoritmo *odd/even exchange sort* (también llamado *odd/even transposition sort*). Asuma que hay *n* procesos P[1..n] y que *n* es par. En este algoritmo cada proceso ejecuta una serie de rondas. En las rondas impares, los procesos con número impar P[impar] intercambian valores con P[impar+1] si los valores están desordenados. En las rondas con número par, los procesos con número par P[par] intercambian valores con P[par+1] si los valores están desordenados (en las rondas pares P[1] y P[*n*] no hacen nada).

a) Determine cuántas rondas deben ejecutarse en el pero caso para ordenar los *n* números.

b) Escriba (utilizando algún mecanismo de memoria compartida) un algoritmo data parallel para ordenar un arreglo de enteros a[1:n] en forma ascendente

c) Modifique el algoritmo para terminar tan pronto como el arreglo fue ordenado
d) Modifique la respuesta de a) para usar *k* procesos, asuma que *n* es múltiplo de *k*

5. Describa en qué consisten los mecanismos de RPC y Rendezvous. Para qué tipo de problemas son más adecuados?

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

7. a) Cuál es el objetivo de la programación paralela?
c) Defina las métricas de speedup y eficiencia. Cuál es el significado de cada una de ellas (que miden)? Ejemplifique.
d) Suponga que el tiempo de ejecución de un algoritmo secuencial es de 10000 unidades de tiempo, de las cuales sólo el 80% corresponden a código paralelizable. Cuál es el límite en mejora que puede obtenerse paralelizando el algoritmo?

8. Sea la siguiente solución al problema del producto de matrices de nxn con P procesos trabajando en paralelo.

```
process worker[w = 1 to P] { # strips en paralelo (p strips de n/P filas) }
        int first = (w-1) * n/P;    # Primera fila del strip
        int last = first + n/P - 1;# Ultima fila del strip
        for [i = first to last] {
            for [j = 0 to n-1] {
                c[i,j] = 0.0;
                for [k = 0 to n-1]
                    c[i,j] = c[i,j] + a[i,k]*b[k,j];
            }
        }
}
```

a) Suponga que n=256 y cada procesador es capaz de ejecutar un proceso. Cuántas asignaciones, sumas y productos se hacen secuencialmente (caso en que P=1)? Cuántas se realizan en cada procesador en la solución paralela con P=8?
b) Si los procesadores P1 a P7 son iguales, y sus tiempos de asignación son 1, de suma 2 y de producto 3, y si P8 es 4 veces más lento, Cuánto tarda el proceso total concurrente? Cuál es el valor del speedup (Tiempo secuencial/Tiempo paralelo)?. Modifique el código para lograr un mejor speedup.