PROGRAMACION CONCURRENTE 2010 (Planes 2003-2007) – Final – 16/3/2011

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.

Parte A : Definiciones y preguntas conceptuales que debieran ser contestadas brevemente (15 puntos)

1- Defina programa concurrente, programa paralelo y programa distribuido.
2- Defina sincronización entre procesos y mecanismos de sincronización.
3- Defina el concepto de no determinismo. Ejemplifique.
4- En qué consiste la propiedad de "A lo sumo una vez" y qué efecto tiene sobre las sentencias de un prog. concurrente?
5- Por qué las propiedades de vida dependen de la política de scheduling? Cuándo una política de scheduling es fuertemente fair?
6- Describa la técnica de passing the baton.
7- En qué consiste la sincronización barrier? Mencione alguna de las soluciones posibles usando variables compartidas.
8- Qué se entiende x arquitectura de grano grueso? Es más adecuada para programas con mucha o poca comunicación?
9- Qué significa que un problema sea de "exclusión mutua selectiva" (EMS)? El problema de los filósofos es de EMS? Por qué? Si en lugar de 5 filósofos fueran 3, el problema seguiría siendo de EMS? Por qué?
10- 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.

Parte B: Interpretación de código (10 puntos: 3 + 3 + 4)

11- Dado el siguiente programa concurrente con variables compartidas:

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

a) Cuáles de las asignaciones dentro del co cumplen la propiedad de "A lo sumo una vez". Justifique.
b) 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, pero sí todos los casos que resulten significativos.

12- Suponga los siguientes programas concurrentes. Asuma que EOS es un valor especial que indica el fin de la secuencia de mensajes, y que los procesos son iniciados desde el programa principal.

P1

```
chan canal (double)
process Genera {
   int fila, col;    double sum;
   for [fila= 1 to 10000]
      for [col = 1 to 10000]
         send canal (a(fila,col));
   send canal (EOS)    }
```

```
process Acumula {
   double valor, sumT;
   sumT=0;
   receive canal (valor);
   while valor<>EOS {
      sumT = sumT + valor
      receive canal (valor); }
   printf (sumT);
}
```

P2

```
chan canal (double)
process Genera {
   int fila, col;    double sum;
   for [fila= 1 to 10000] {
      sum=0;
      for [col = 1 to 10000]

 sum=sum+a(fila,col);
      send canal (sum);
}
   send canal (EOS)
}
```

```
process Acumula {
   double valor, sumT;
   sumT=0;
   receive canal (valor);
   while valor<>EOS {
      sumT = sumT + valor
      receive canal (valor);
 }
   printf (sumT);
 }
```

a) Qué hacen los programas?

b) Analice desde el punto de vista del número de mensajes.
c) Analice desde el punto de vista de la granularidad de los procesos.
d) Cuál de los programas le parece más adecuado para ejecutar sobre una arquitectura de grano grueso de tipo cluster de PCs? Justifique.

13- 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))*

Parte C: Temas para desarrollar (12 puntos: 3 + 5 + 4)

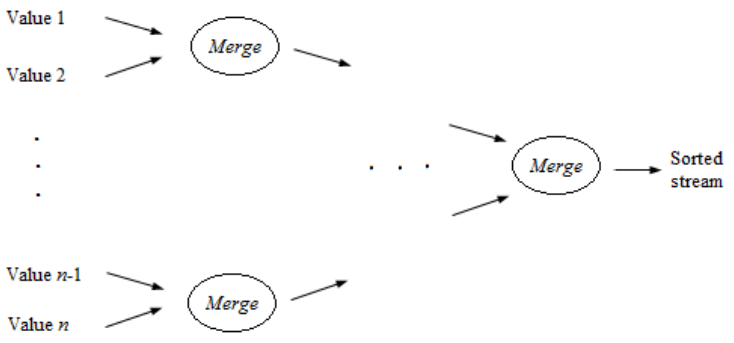
14- a) Describa brevemente en qué consisten los mecanismos de RPC y Rendezvous. Para qué tipo de problemas son más adecuados?
b) Por qué es necesario proveer sincronización dentro de los módulos en RPC? Cómo puede realizarse esta sincronización?
c) Qué elementos de la forma general de rendezvous no se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

15- Explique sintéticamente los 7 paradigmas de interacción entre procesos en programación distribuida. En cada caso ejemplifique, indique qué tipo de comunicación por mensajes es más conveniente. Justifique sus respuestas.

16- a) Cuál es el objetivo de la programación paralela?
b) Mencione al menos 4 problemas en los cuales Ud. entiende que es conveniente el uso de técnicas de programación paralela.
c) Defina las métricas de speedup y eficiencia. Cuál es el significado de cada una de ellas (qué miden)? Cuál es el rango de valores posibles de cada uno? Ejemplifique.
d) Suponga que la solución a un problema es paralelizada sobre *p* procesadores de dos maneras diferentes. En un caso, el speedup (S) está regido por la función S=p/3 y en el otro por la función S=p-3. Cuál de las dos soluciones se comportará más eficientemente al crecer la cantidad de procesadores? Justifique claramente.
e) Suponga que el tiempo de ejecución de un algoritmo secuencial es de 10000 unidades de tiempo, de las cuales sólo el 80% corresponde a código paralelizable. Cuál es el límite en mejora que puede obtenerse paralelizando el algoritmo? Justifique

Parte D: Ejercicio a resolver (8 puntos)

17- 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.

TOTAL = 45 puntos. Para aprobar se requieren 27 puntos, correspondientes al menos a 3 de las partes, una de las cuales debe ser la parte D