PROGRAMACION CONCURRENTE - EXAMEN FINAL 16-6-2010

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. a) Defina programa concurrente, programa paralelo y programa distribuido.
b) Compare la comunicación x mensajes sincrónicos y asincrónicos en cuanto al grado de concurrencia y posibilidad de entrar en deadlock.
c) Defina el concepto de "continuidad conversacional" entre procesos

2. 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 tipo cluster de PCs? Justifique.

3. a) Describa la técnica de "passing the baton"? Cuál es su utilidad? Ejemplifique.
b) Qué relación encuentra con la técnica "passing the condition"? Ejemplifique.

4. 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

```
...
int cant=1000;
DO   (cant < -10);  datos?(cant) ⟶
Sentencias 1
     □ (cant > 10);  datos?(cant) ⟶
Sentencias2
     □ (INCOGNITA);  datos?(cant) ⟶
Sentencias3
END DO
...
```

Segmento 2

```
...
int cant=1000;
While (true)
{ IF   (cant < -10);  datos?(cant) ⟶
Sentencias 1
     □ (cant > 10);  datos?(cant) ⟶
Sentencias2
     □ (INCOGNITA);  datos?(cant) ⟶
Sentencias3
   END IF
}
...
```

a) *INCOGNITA* equivale a: ***(cant = 0),***

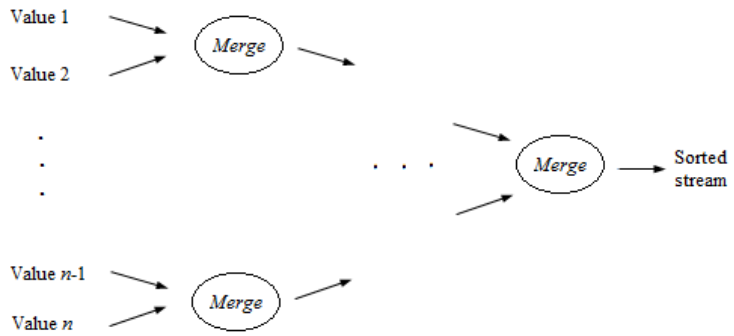
b) *INCOGNITA* equivale a: *(cant > -100)*
c) *INCOGNITA* equivale a: *((cant > 0) or (cant < 0))*
d) *INCOGNITA* equivale a: *((cant > -10) or (cant < 10))*
e) *INCOGNITA* equivale a: *((cant >= -10) or (cant <= 10))*

5. 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.

6. a) Cuál es el objetivo de la programación paralela?
b) Defina las métricas de speedup y eficiencia. Cuál es el significado de cada una de ellas (qué miden) y su rango de valores? Ejemplifique.
c) 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-1 y en el otro por la función S=p/2. Cuál de las dos soluciones se comportará más eficientemente al crecer la cantidad de procesadores? Justifique claramente.
d) Suponga que el tiempo de ejecución de un algoritmo secuencial es de 10000 unidades de tiempo, de las cuales sólo el 90% corresponde a código paralelizable. Cuál es el límite en mejora que puede obtenerse paralelizando el algoritmo? Justifique

7. Describa los mecanismos de comunicación y sincronización provistos por MPI, Ada, Java y Linda.
8. 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.