Programación Concurrente 2024

Cuestionario guía - Clases Teóricas 1 y 2

1- Mencione al menos 3 ejemplos donde pueda encontrarse concurrencia

2- Escriba una definición de concurrencia. Diferencie procesamiento secuencial, concurrente y paralelo.

3- Describa el concepto de *deadlock* y qué condiciones deben darse para que ocurra.

4- Defina inanición. Ejemplifique.

5- ¿Qué entiende por *no determinismo*? ¿Cómo se aplica este concepto a la ejecución concurrente?

6- Defina comunicación. Explique los mecanismos de comunicación que conozca.

7- a) Defina sincronización. Explique los mecanismos de sincronización que conozca.
b) ¿En un programa concurrente pueden estar presentes más de un mecanismo de sincronización? En caso afirmativo, ejemplifique

8- ¿Qué significa el problema de "interferencia" en programación concurrente? ¿Cómo puede evitarse?

9- ¿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.

10- Dado el siguiente programa concurrente:

```
x = 2; y = 4; z = 3;
co
  x = y - z //  z = x * 2  // y = y - 1
oc
```

a) ¿Cuáles de las asignaciones dentro de la sentencia *co* cumplen con ASV?. Justifique claramente.
b) Indique los resultados posibles de la ejecución

Nota 1: las instrucciones NO SON atómicas.
Nota 2: no es necesario que liste TODOS los resultados, pero si los que sean representativos de las diferentes situaciones que pueden darse.

11- Defina acciones atómicas condicionales e incondicionales. Ejemplifique.

12- Defina propiedad de seguridad y propiedad de vida.

13- ¿Qué es una política de scheduling? Relacione con fairness. ¿Qué tipos de fairnes conoce?

14- ¿Por qué las propiedades de vida dependen de la política de scheduling? ¿Cómo aplicaría el concepto de fairness al acceso a una base de datos compartida por n procesos concurrentes?

15- Dado el siguiente programa concurrente, indique cuál es la respuesta correcta (justifique claramente)

```
int a = 1, b = 0;
co ⟨await (b = 1) a = 0 ⟩  // while (a = 1) { b = 1; b = 0; }  oc
```

a) Siempre termina
b) Nunca termina
c) Puede terminar o no