Grado en Ingeniería Informática
Programación Concurrente y Tiempo Real
Problemas de Algoritmos con Memoria Común

1. Considerar el siguiente fragmento de programa para dos procesos A y B:

   ```
   Proceso A Proceso B
   for a:=1 to 10  for b:= 1 to 10
     x:=x+1;     x:=x+1;
   ```

   Suponer que la variable `x` está inicializada a cero, que ambos procesos se ejecutan una sola vez y que las variables `a` y `b` no son compartidas. Los dos procesos pueden ejecutarse a cualquier velocidad. ¿Cuáles son los posibles valores resultantes para `x`? Para cada valor, indique la secuencia de entrelazado que lleva a él. Nota: suponga que `x` debe ser cargada en un registro para incrementarse.

2. En vez de la instrucción `Test_and_Set`, algunos ordenadores proveen de una instrucción atómica que incrementa en 1 el valor de una variable `Lock`:

   ```
   Function {ATOMICA} TestAndInc (Var Lock: Integer):Integer;
   Begin
     TestAndInc:=Lock;
     Lock:=Lock+1;
   End;
   ```

   Escribir protocolos de entrada y salida para proteger una sección crítica utilizando dicha solución.

3. Escribir un programa concurrente para multiplicar matrices de $n \times n$ con n procesos concurrentes. Suponga que los elementos de las matrices se pueden leer simultáneamente. Extienda el programa para realizar concurrentemente el cálculo del producto de ambas diagonales de una matriz.

4. Escribir un programa para calcular el producto escalar de dos vectores de n componentes. Puede utilizar tantos procesos concurrentes como sea necesario.

5. ¿Qué diferencia fundamental existe entre un bucle de espera activa y una variable interruptor que da turnos para la sincronización de procesos?

6. ¿Utiliza la espera ocupada el Algoritmo de Dekker, si el segundo proceso está en su sección crítica, `Turno=1` y el primer proceso está intentando entrar en su sección crítica?

7. Probar que el Algoritmo de Dekker hace imposible que un proceso espere indefinidamente a entrar en su sección crítica, con la suposición de que siempre que un proceso entra en la sección crítica eventualmente la abandonará.

8. Generalizar el Algoritmo de Dekker a tres procesos.

9. ¿Qué ocurriría si el Algoritmo de Dekker se hubiera programado así?

```
Procedure Pi;
Begin
  Repeat
    Turno := i;
    While Turno<>i Do;
        Sección_Crítica;
    Turno := j;
  Forever;
End;
```

10. Al siguiente algoritmo se le conoce como solución de Hyman al problema de la exclusión mutua. ¿es correcta dicha solución?

```
Proceso 1 Proceso 2
c1:=0; c2:=0;
while turno<>1 do while turno<>2 do
begin begin
  while c2=0; while c1=0 do
  turno:=1;   turno:=2;
end; end;
Sección_Crítica; Sección_Crítica;
c1:=1; c2:=1;
```

Inicialmente, las variables c2,c2 y turno valen 1.

11. La instrucción EX intercambia los contenidos de dos posiciones de memoria, y es equivalente a la ejecución atómica e indivisible de las tres operaciones siguientes:

```
temp :=a;
a:=b;
b:=temp;
```

Construya un mecanismo de exclusión mutua basado en la instrucción EX.

12. El siguiente programa es una solución al problema de la exclusión mutua para dos procesos. Discutir la corrección de la solución; si es correcta, probarlo formalmente y si no lo es, proponer una secuencia de entrelazado que lleve a romper la exclusión mutua.

```
Program Prueba;
Var c1, c2: Integer;

Procedure P1 Procedure P2;
Begin   Begin
Repeat    Repeat
    Resto_código   Resto_código;
      Repeat       Repeat
c1:=1-c2;   c2:=1-c1;
Until c2<>0   Until c1<>0;
    Crítica;      Crítica;
    c1:=1;     c2:=1;
  Forever   Forever;
End; End;

Begin
  c1:=1;
  c2:=1;
  cobegin
    P1;P2
  coend;
End;
```

13. Si se quiere pasar un semáforo como parámetro, ¿cómo habría que pasarlo, por valor o por referencia? ¿Qué ocurre si se escoge la opción equivocada?

14. Añadir un semáforo al siguiente programa de modo que siempre imprima 40.

```
Program Aumentar;
Const m=20;
Var n: Integer;

Procedure inc;
Var i :Integer;
Begin
  For y:=1 to m Do
    n:=n+1;
End;

Begin {Principal}
  n:=0;
  Cobegin
    inc;inc
  Coend;
  Writeln (n);
End.
```

15. ¿Funcionaría una solución de espera ocupada basada en una variable de turno si los dos procesos se ejecutaran en paralelo en dos procesadores distintos, pero con un esquema de memoria común?

16. ¿Qué cualidad han de poseer las operaciones de los semáforos para que el semáforo funcione? ¿Cómo se puede conseguir dicha cualidad al implementar semáforos en el sistema operativo?

17. Considérese un programa concurrente con las tareas P y Q, que se muestran a continuación. A, B, C, D y E con sentencias arbitrarias atómicas (indivisibles).

```
Task body P is      Task body Q is
 begin                begin
   A;                   D;
   B;                   E;
   C;                 end;
 end;
```

    Indicar todos las posibles entrelazados de la ejecución de los dos procesos anteriores (indicarlo por medio de "trazas"de ejecución dadas en términos de sentencias atómicas).

18. ¿Qué diferencia fundamental existe entre un bucle de espera activa y una variable interruptor que da turno para la sincronización de procesos?

19. El siguiente código debe imprimir como salida 50 ó 110 ¿Lo hace correctamente?

```
Program Incógnita

Var x: integer;
Task Body P1 is          Task Body P2 is
Begin                    Begin
  x:=x+10;               If x>100 then write(x) else write(x-50)
End P1;                  End P2;

Begin
  Cobegin
    P1;P2;
  Coend;
End Incógnita.
```