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.

Los posibles valores para la variable (compartida) x es de 10 – 20.

Para que se produzca un valor < 20, se tiene que producir un entrelazado entre las operaciones de incremento. Recordemos que la operación de incremento a nivel de maquina requiere de: $temp = LOAD(x) (cargar el valor de x), $temp = $temp +1 (sumar el inmediato 1 al valor x y guardarlo), STORE(x,$temp) (guardar el nuevo valor de x). Teniendo todo esto en cuenta:
Si partimos desde el principio (x = 0), y tanto el proceso (ligero o hebra) A como el B, realizan la carga del valor de x (LOAD(x)) y realizan también, de forma simultánea, las operaciones de incremento y de guardado, se habrá perdido uno de los incrementos debido a que tanto A como B han escrito en el valor x 1. Nótese que esto podría ocurrir en cualquiera de los momentos (las 10 veces) en los que se incrementa x. Si pasara siempre obtendríamos 10, si pasara nunca obtendríamos 20, si pasara 1 vez obtendríamos 19 y así con el resto.

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.

Procedure increment_task
Begin

    int n := 0;
    for a:= 0 to 10

        TestAndInc(n);

End;

(Creo que es así, a confirmar por Juan Carlos).

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.

```
Procedure principal
Begin
    int n;
    int M1[n][n];
    int M2[n][n];
    int P[n][n];
    cobegin
        mulMat(0); mulMat(1); … mulMat(n-1);
    coend;
End;

Procedure mulMat
Begin
    int i; //Numero recibido
    for j := 0 to n
        for k :0 to n
            P[i][j] = P[i][j] + (M1[i][k] * M2[k][j]);
End;
```

Hazte una traza y veras que sale. ;)

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

```
Procedure principal
Begin
    int n;
    int V1[n];
    int V2[n];
    int resultado := 0;
    int resultadosParciales[n];
    cobegin
        prodVec(0); prodVec(1); … prodVec(n-1);
    coend;
    for i:= 0 to n
        resultado := resultado + resultadosParciales[i]
End;


Procedure prodVec
Begin
    int i; //Numero recibido
    resultadosParciales[i] = V1[i]*V2[i]
```

Creo que así vale.

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