PRÁCTICO 4

Objetivos

- Comprender el problema de la mutua exclusión y las dificultades de probar la correctitud de programas concurrentes
- Ver soluciones por software y por hardware al problema de la mutua exclusión

Ejercicio 1 (básico)

Escribir el grafo de ejecución y un programa que evalúe la siguiente expresión, usando cobegin-coend y de modo de alcanzar el mayor grado posible de concurrencia.

$$3 \times a \times b + 4 / ((c + d)^{e-f})$$

La evaluación de una expresión sólo debe esperar por la evaluación de sus subexpresiones.

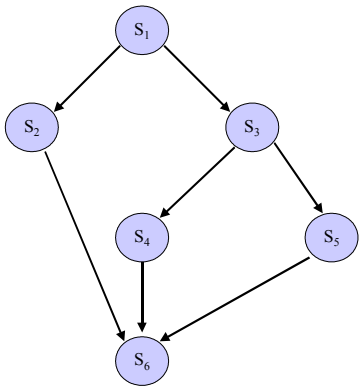
Ejercicio 2 (básico)

Construir un programa que usando cobegin-coend calcule el producto de una matriz de 3×3 por otra de 3×2 con 6 procesos concurrentes.

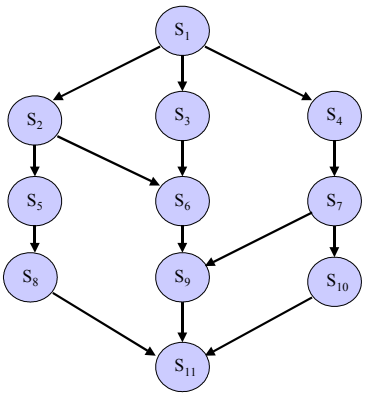
Ejercicio 3 (básico)

Dados los siguientes grafos de precedencia, ¿pueden representarse con cobegin-coend? Sino mostrar por qué.

a)



b)



Ejercicio 4 (en clase)

Se desea contar el número de veces que es ejecutado un determinado proceso y para esto se definen dos contadores globales:

```
var unidades, decenas : integer := 0
```

que son usados de la siguiente forma:

```
procedure actividad is
begin
  for i in 1..12 loop
    actividad_propia_del_proceso
    unidades := unidades + 1;
    if (unidades = 10) then
      unidades := 0;
      decenas := decenas + 1;
    end if;
  end loop;
end procedure;
```

Se ejecutan **concurrentemente** dos copias de actividad:

- Analizar la conducta del algoritmo e indicar algunos los resultados posibles de los contadores.
- Indicar cómo podría obtenerse siempre el resultado correcto.

Ejercicio 5 (medio)

Implementar usando el algoritmo de Dekker una solución al problema de concurrencia planteado en el **Ejercicio 4**.

Ejercicio 6 (avanzado)

1. Realizar un análisis completo del algoritmo de Dekker a los efectos de demostrar que resuelve correctamente el problema de la mutua exclusión
2. ¿Qué desventajas tiene el algoritmo?

Ejercicio 7 (medio)

Dado el siguiente programa:

```
program prueba_incremento;
  var n : integer;

  procedure incremento;
    var i : integer;

  begin
    for i in 1..20 loop
      n := n + 1;
  end procedure;
```

```
begin { program }
  n := 0;
  cobegin
    incremento;
    incremento;
  coend;
  writeln('la suma es : ', n)
end program.
```

Discutir los posibles resultados en las siguientes situaciones:

1. La máquina tiene un solo procesador y la construcción n:=n+1 se ejecuta en **una sola** instrucción de máquina.
2. La máquina tiene un solo procesador y la construcción n:=n+1 se ejecuta en **más de una** instrucción de máquina.

Ejercicio 8 (medio)

En algunos procesadores existe la instrucción de máquina ex(c,a) que es equivalente a la ejecución indivisible de:

```
  temp := c;
  c := a;
  a := temp;
```

siendo temp una variable interna temporal ajena al programa.

1. Discutir la correctitud de la siguiente solución al problema de la mutua exclusión entre dos procesos.

```
program exchange;
  var c : integer;

  procedure p1;
    var a1 : integer;
  begin
    a1 := 0;
    repeat
      sin_importancia1;
      repeat
        ex(c,a1)
      until a1 = 1;
      sec_crit1;
      ex(c,a1)
    forever
  end procedure;
procedure p2;
    var a2 : integer;
  begin
    a2 := 0;
    repeat
      sin_importancia2;
      repeat
        ex(c,a2)
      until a2 = 1;
```

```
      sec_crit2;
      ex(c,a2)
    forever
  end procedure;

begin {programa principal}
  c := 1;
  cobegin
    p1;
    p2;
  coend
end program.
```

2. ¿Qué ocurriría si la instrucción ex(c,a) se reemplazara por las tres asignaciones siguientes?

```
  temp := c;
  c := a;
  a := temp;
```

Ejercicio 9 (medio)

Supóngase la instrucción tst(a,c) que equivale a la ejecución indivisible de:

```
  a := c;
  c := 1;
```

donde c es una variable global.

1. Discutir la correctitud de la siguiente solución al problema de la mutua exclusión para 2 procesos.

```
program test_and_set;
  var c : integer;

  procedure p1;
    var a1 : integer;
  begin
    repeat
      sin_importancia1;
      repeat
        tst(a1, c)
      until (a1 = 0);
      seccion_critica_1;
      c := 0;
    forever;
  end procedure;

  procedure p2;
    var a2 : integer;
  begin
    repeat
      sin_importancia2;
      repeat
        tst(a2, c)
```

```
      until (a2 = 0);
      seccion_critica_2;
      c := 0;
    forever;
  end procedure;

begin
  c := 0;
  cobegin
    p1;
    p2;
  coend
end program.
```

2. Generalizar para "n" procesos. ¿Qué cambios se necesitan?

3. Indicar que ocurriría si la instrucción tst(a,c) se reemplazara por las dos asignaciones

```
a := c;
c := 1;
```