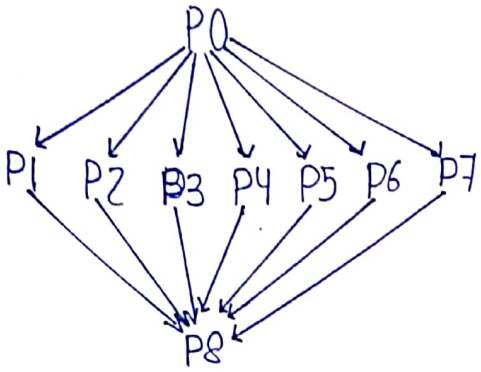
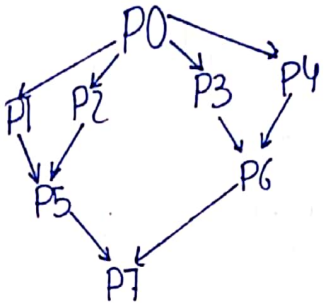
4)





5)

(1) Incremento impulsos (2) Impresión del contador (3) Reestablece el contador

Secuencia correcta

~~(1) Incrementa impulsos (2) (3)~~

(2) Impresión del contador (3) Reestablece el contador (1) Aumento impulsos

Secuencia correcta, el nuevo impulso se cuenta en la hora siguiente

(2) Impresión del contador (1) Aumento impulso (3) Reestablece el contador

Secuencia incorrecta, se ~~pue~~ pierde un impulso

6) $T_s(2n) = S = 2n^2 - n$

a) Las dos instancias concurrentes de sort se ejecutan en el mismo procesador

$$P2 = 2 \cdot T_s(n) = \left(\frac{n^2}{2} - \frac{n}{2}\right) \cdot 2 = n^2 - n$$

b) Cada instancia de Sort se ejecuta en un procesador distinto

Sort $(1, n)$ tarda $T_s(n) = \frac{n^2 - n}{2}$

Sort $(n+1, 2n)$ tarda $T_s(n) = \frac{n^2 - n}{2}$

P2 será el mayor de los dos tiempos anteriores, pero al ser iguales

$$P2 = \frac{n^2 - n}{2}$$

6) Comparación Cualitativa:

Añado el tiempo de Merge a P1 y P2

$P1 = n^2 - n + 2n = n^2 + n.$

$P2 = \frac{n^2-n}{2} + 2n = \frac{n^2-n+4n}{2} = \frac{n^2+3n}{2}$

Veamos si P1 es mejor que S

$(2n^2-n) - (n^2+n) > 0 \Rightarrow n^2 - 2n > 0 \Rightarrow n\cdot(n-2) > 0 \Leftrightarrow n > 2$

luego si $n > 2$ P1 mejor que S

Veamos si P2 mejor que S

$$(2n^2-n) - \left(\frac{n^2+3n}{2}\right) > 0 \Rightarrow \frac{4n^2-2n-n^2-3n}{2} > 0 \Rightarrow \frac{3n^2-5n}{2} > 0 \Rightarrow$$

$$\Rightarrow 3n^2 - 5n > 0 \Rightarrow n\cdot(3n-5) > 0 \Leftrightarrow n > \frac{5}{3}$$

luego si $n \geq 2$ P2 es mejor que S (suponiendo n entero positivo)

Veamos que P2 es mejor que P1

$$(n^2+n) - \left(\frac{n^2+3n}{2}\right) > 0 \Rightarrow \frac{2n^2+2n-n^2-3n}{2} > 0 \Rightarrow n^2 - n > 0 \Leftrightarrow n > 1$$

luego P2 es siempre mejor que P1, pues no tiene sentido un n nulo.

7)

```
Var a,b,c: array [1...3,1...3] of real;
process Producto3procesos ()
begin
   Multiplicar_Conc (1);
   Multiplicar_Conc (2);
   Multiplicar_Conc (3);
coend
end

procedure Multiplicar_Concu(i: 1...3)
   var j,k :integer;
begin
      for j:=1 to 3 do begin
         c[i,j]:=0;
         for k:=1 to 3 do
          c[i,j] := c[i,j] + a[i,k] * b[k,j];
      end
end
```

8)

[Variable compartida]
Var serial : array[1...9] of boolean := (false, ..., false);

```
procedure Acabar(i: integer)
begin
  while not Prodi[i] do
```

```
procedure Esperarpor (i: integer)
  begin
    while serial[i] == false do begin
  end
end
```

```
procedure Acabar (i: integer)
  begin
    if i == 9 then
      for j := 1 to 9 do
        serial[j] := false
    else
      serial[i] := true
  end
end
```

9)

```
process P[1]
  begin
    S1;
    Acabar(1);
  end
end
```

```
process P[2]
  begin
    Esperarpor(1);
    S2;
    Acabar(2);
  end
end
```

```
process P[3]
  begin
    Esperarpor(2);
    S3
    Acabar(3)
  end
end
```

```
process P[4]
  begin
    Esperarpor(1);
    S4
    Acabar(4)
  end
end
```

```
process P[5]
  begin
    Esperarpor(2);
    Esperar por(4);
    S5
    Acabar(5)
  end
end
```

```
process P[6]
  begin
    Esperarpor(3);
    Esperar por(5);
    S6
    Acabar(6)
  end
end
```

```
process P[7]
  begin
    Esperarpor(4);
    S7
    Acabar(7);
  end
end
```

```
process P[8]
  begin
    Esperarpor(5);
    Esperarpor(7);
    S8;
    Acabar(8);
  end
end
```

```
process P[9]
  begin
    Esperarpor(6);
    Esperarpor(8);
    S9;
    Acabar(9);
  end
end
```