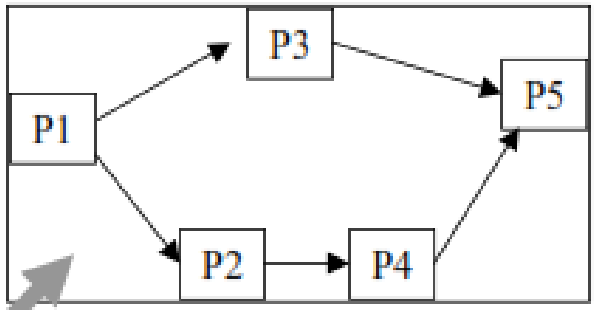
# Ejercicios

- **Ejercicio 1:** Un grafo de precedencias de tareas es un grafo dirigido acíclico que establece un orden de ejecución de tareas de un programa concurrente, de manera que una tarea no puede empezar a ejecutarse hasta que las que sean anteriores en el grafo hayan terminado. Por ejemplo, si consideramos el grafo de la figura 1, la tarea P4 no puede empezar a ejecutarse hasta que P2 y P1 terminen, mientras que P5 no puede empezar hasta que P1, P2, P3 y P4 hayan terminado. Asumamos que cada tarea ejecuta el siguiente código:

```
espera a que las tareas anteriores terminen
ejecuta el cuerpo de la tarea
avisa de su terminación a quien corresponda
```




1) Usando semáforos, escribir el programa concurrente correspondiente al diagrama de la figura

2) Esquematizar un método general de sincronización para un grafo de precedencias general. Calcular el número de semáforos que el método utilizaría. Este número se puede poner, por ejemplo, como función del número de tareas, de arcos, etc.

# Ejercicios

- **Ejercicio 2**: Considerar el siguiente programa concurrente:

  Calcular para él el conjunto de los posibles valores finales para la variable x

```
Vars x: Ent := 0;
     s1: sem := 1;
     s2: sem := 0
P1::          | P2::          | P3::
              |               |
wait(s2)      | wait(s1)      | wait(s1)
wait(s1)      | x := x*x      | x := x+3
x := 2*x      | send(s1)      | send(s2)
send(s1)      |               | send(s1)
```

- **Ejercicio 3:** Implementar un semáforo general en base a uno (o varios) semáforo binarios

# Ejercicios

- **Ejercicio 4**: Problema de uso de recursos: **n** procesos compiten por el uso de un recurso, del que se disponen de k unidades. Las peticiones de uso del recurso son del tipo:

  ***reserva(r)***: --necesito se me concedan, "de golpe", r unidades del recurso

  ***libera(l)***: --libero, "de golpe", l unidades del recurso, que previamente se me habían concedido

- Programar dichas operaciones usando semáforos binarios

# Ejercicios

- **Ejercicio 5**: Considerar el siguiente programa, que presenta un problema: nada impide que se use una variable en una expresión con un valor todavía indefinido.
- Se pide lo siguiente. Utilizando semáforos, completar el código de dicho programa de manera que nunca una variable sea usada en una expresión antes de que se le haya asignado un valor, ya sea mediante una instrucción `leer` o una asignación.
- Explicar cómo se generalizaría dicha solución para cualquier programa. ¿Podría un programa escrito de acuerdo a la propuesta anterior llegar a bloquearse? Si es así, escribir un ejemplo sencillo de bloqueo. Si no es posible, justificar "de manera convincente" por qué es así.

```
Vars a,b,c,d: Ent;
P1::
   leer(a);
   c := a+b
   b := 2*d
P2::
   leer(c)
   leer(d)
   b := a+d+c
```