1. Puertos con celdas:
   TAD = Tipo Abstracto de Datos.

Requisito	Codigo
Encapsulamiento del estado interno	TailCell es una celda privada dentro de NuevoPuerto. Solo el procedimiento P puede accederla.
Acceso únicamente mediante operaciones públicas	Solo puedes modificar el flujo usando el procedimiento P (que se devuelve como puerto) y la operación pública Enviar.
Persistencia del estado	TailCell mantiene el final actual del flujo a través del tiempo. Cada Enviar modifica el mismo flujo.
Evolución controlada del estado	Cada vez que llamas {Enviar P X}, el procedimiento P actualiza TailCell de forma controlada.
Interfaz clara y bien definida	Las operaciones públicas son claramente {NuevoPuerto} y {Enviar}; el usuario no toca directamente el flujo.
Abstracción de la implementación interna	El usuario no sabe que internamente existe un TailCell ni cómo se extiende el flujo. Solo ve que puede {Enviar} mensajes y observar el flujo.

2. Celdas con puertos:

Requisito	Codigo
Encapsulamiento del estado interno	EstadoActual está local dentro de Procesar. No se puede acceder directamente desde afuera.
Acceso únicamente mediante operaciones públicas	Solo puedes interactuar usando {Acceder}, {Asignar}, {NuevaCelda}. No existe otra forma de tocar el estado.
Persistencia del estado	El valor en EstadoActual persiste entre operaciones gracias al hilo {Procesar X Flujo} que mantiene vivo el estado.
Evolución controlada del estado	Solo cambia mediante {Asignar}, que envía un mensaje asignar(NuevoValor PuertoRespuesta) al hilo que actualiza EstadoActual.
Seguridad concurrente	El thread procesa uno a uno los mensajes (acceder, asignar), garantizando que no haya conflictos de concurrencia.

Requisito	Codigo
Interfaz clara y bien definida	Las operaciones {NuevaCelda}, {Acceder}, {Asignar} definen claramente lo que se puede hacer con la celda.
Abstracción de la implementación	El usuario no necesita saber que internamente hay un flujo, un puerto o un thread. Solo usa {Acceder} y {Asignar} normalmente.

3. La figura ilustra una posible salida para las entradas dadas por los clientes. Sin embargo, esta no es la única salida posible. ¿Por qué? ¿Cuáles son las otras posibles salidas para la misma entrada?
   Respuesta: Porque en programación concurrente por paso de mensajes el orden en que los mensajes llegan al servidor (contador) NO está garantizado.
   Otras posibles salidas:

```
[a#1]|[a#2]|[a#2 b#1]|[a#2 b#1 c#1]|[a#2 b#1 c#2]|_
[a#1]|[a#1 b#1]|[a#2 b#1]|[a#2 b#1 c#1]|[a#2 b#1 c#2]|_
[a#1]|[a#1 b#1]|[a#2 b#1]|[a#2 b#1 c#1]|[a#2 b#1 c#2]|_
[a#1]|[a#1 c#1]|[a#2 c#1]|[a#2 c#1 b#1]|[a#2 c#2 b#1]|_
[a#1]|[a#2]|[a#2 b#1]|[a#2 b#1 c#1]|[a#2 b#1 c#2]|_
[a#1]|[a#1 c#1]|[a#2 c#1]|[a#2 c#1 b#1]|[a#2 c#2 b#1]|_
```

4. Se definieron objetos puerto reactivos Portero y Counter; y no reactivos Logger y se logró la interacción entre ambos reactivos y no reactivos