1. Utilizando una región de memoria compartida, implemente una cola circular y utilice un semáforo para la sincronización de acceso, otro para mantener la cantidad de entradas libres y como recurso de bloqueo para procesos productores y un tercero para mantener la cantidad de elementos en la cola y como recurso de bloqueo para procesos consumidores.
   Implemente los servicios:

```
// Inicializa (debe residir en un segmento de shared memory)
void QueueInit(Queue_t *pQ);

// Desstruye el contenedor, liberando recursos
void QueueDestroy(Queue_t *pQ);

// Agrega un Nuevo elemento. Bloquea si no hay espacio
void QueuePut(Queue_t *pQ, int elem);

// Remueve y retorna un elemento, bloquea si no hay elementos
int QueueGet(Queue_t *pQ);

// recupera la cantidad de elementos en la cola
Int QueueDixr(Queue_t *pQ);
```

2. Utilizando los servicios implementados en el problema 1, implemente:
   a. Un proceso que cree e inicialize la cola circular.
   b. Un proceso que agregue elementos a la cola
   c. Un proceso que remueva elementos
   d. Un proceso que monitoree el estado de la cola.

3. Que protocolo implementaría para evite las “carreras” que se presentan en el problema anterior, si todos los procesos son arrancados “simultáneamente”.
4. Como trataría de evitar que una salida inesperada de uno de los procesos (por ejemplo un Ctrl-C) cree un deadlock en el sistema?