14. Considere la siguiente implementación para el problema de los lectores-escritores utilizando monitores para la implementación de las funciones para acceder a los datos compartidos.

    Considere que si c es una variable de tipo *condition*, entonces la función booleana $empty(c)$ retorna $falso$ cuando existan uno o varios procesos esperando en la cola de c, de lo contrario retorna *verdadero*.

    a) Explique las políticas de prioridad utilizadas para los lectores y escritores para el diseño de esta solución.

    b) ¿Existe la posibilidad de inanición?

```
MONITOR lector-escritor{
    int lecto_cant, escribiendo = 0;
    condition ok_leer, ok_escribir;
INICIO-LEER{
    if (escribiendo || !empty(ok_escribir)) ok_leer.wait;
    lecto_cant = lecto_cant + 1;
    ok_leer.signal;
}
FIN-LEER {
    lecto_cant = lecto_cant - 1;
    if (lecto_cant == 0) ok_escribir.signal;
}
INICIO-ESCRIBIR {
    if ((lecto_cant != 0) || writing) ok_escribir.wait;
    escribiendo = 1;
}
FIN-ESCRIBIR {
    escribiendo = 0;
    if (!empty(ok_leer)) ok_leer.signal;
    else ok_escribir.signal;
}
}
```

15. Escriba un algoritmo que resuelva el problema productor-consumidor, con un tamaño de buffer limitado, utilizando monitores para la sincronización y exclusión.