```
#---------Opción A--------------#

fun esta?(letra, pila):
    si vacia?(pila) entonces
        devolver False
    fin si
    si mayuscula(letra) == mayuscula(cima(pila)) entonces
        devolver True
    si no:
        devolver esta?(letra, desapilar(pila))
    fin si
fin fun


fun esta_todas(p1, p2):
    si vacia?(p1) o vacia?(p2) entonces
        devolver False
    fin si
    aux = True
    pAux = NULL  # Copia temporal de p1 para no modificarla
    mientras !vacia?(p1):
        si !esta?(cima(p1), p2) entonces
            aux = False
        fin si
        apilar(cima(p1), pAux)
        desapilar(p1)
    fin mientras

    # Restaurar p1 a partir de pAux
    mientras !vacia?(pAux):
        apilar(cima(pAux), p1)
        desapilar(pAux)
    fin mientras

    devolver aux
fin fun


fun esta_igual(p1, p2):
    si vacia?(p1) y vacia?(p2) entonces
        devolver True
    fin si

    si (vacia?(p1) y !vacia?(p2)) o (!vacia?(p1) y vacia?(p2)) entonces
        devolver False
    fin si

    # Verificar si todas las letras de p1 están en p2
    si !esta_todas(p1, p2) entonces
        devolver False
    fin si

    # Verificar si la cima de p1 y p2 son iguales ignorando mayúsculas/minúsculas
    si mayuscula(cima(p1)) != mayuscula(cima(p2)) entonces
        devolver False
    si no:
        # Recursión con el resto de las pilas
        devolver esta_igual(desapilar(p1), desapilar(p2))
    fin si
fin fun

#---------------Ejercicio 3------------------#
tipo sistema:
    cola_criticos = c_criticos
    cola_no_criticos = c_NO

#----------a---------#
fun crea_s():
    sistema = crear();
    sistema.c_criticos = crear();
    sistemas.c_NO = crear();
    dev sistema
fin fun

#-------------b-----------#
fun añadir_mensaje(s, mensaje):
    si es_critico?():
        encolar(s.c_criticos, mensaje)
    si no:
        encolar(s.c_NO, mensaje)
    fin si
    dev s
fin fun

#--------------c-----------#
fun primer_mensaje(s):
    si es_vacia?(s.c_criticos) y es_vacia?(s.c_NO):
```

```
        dev NULL
    fin si

    si !es_vacia?(s.c_criticos): #prioridad para la cola critica
        dev cima(s.c_criticos)
    si no y !es_vacia(s.c_NO):
        dev cima(s.c_NO)
    fin si
fin fun

#------------------d-------------#
fun solo_criticos(cola):
    si es_vacia?(cola): dev NULL
    fin si

    cola_Critica = NULL
    mientras !es_vacia(cola):
        si es_critico(cima(cola)):
            encolar(cima(cola), cola_Critica)
        fin si
        desencolar (cola)
    fin mientras
    dev cola_Critica
fin fun

#---------e----------#
fun llevar_al_final(cola):
    si es_vacia?(cola):
        dev cola
    fin si

    mensaje = cima(cola)
    desencolar(cola)

    si es_critico?(mensaje) entonces
        encolar(cola, mensaje)
    si no:
        cola = llevar_al_final(cola)
        encolar(cola, mensaje)
    fin si

    dev cola
fin fun
#----------------f--------------#
fun numero_criticos(sistema):
    si es_Vacio?(sistema.cola_criticos):
        dev 0
    si no:
        desencolar(sistema.cola_criticos)
        dev 1 + numero_criticos(sistema)
    fin si
fin fun

#--------------g----------#
fun sistema_vacio(sistema):
    dev (es_Vacio?(sistema.cola_criticos) y es_Vacio?(sistema.cola_NO))
fin fun

#el h está implementado directamente
```