[illegible]

Parte 1 : Seguridad

```
Monitor _ C _ P

    ncn : int = 0      : nº coches norte
    ncs : int = 0      : nº coches sur
    np  : int = 0      : nº peatones
    pueden-norte : VC = True
    pueden-sur   : VC = True
    pueden-peatones : VC = True
    INV : {ncn ≥ 0 , ncs ≥ 0, np ≥ 0 , ncn · ncs = 0 , ncn · np = 0, ncs · np = 0 }

wants-enter-north()
    {INV}
    pueden-norte . wait (ncs == 0 ∧ np == 0)
    ncn += 1
    {INV ∧ ncn > 0}

wants-enter-south()
    {INV}
    pueden-sur-wait (ncn == 0 ∧ np == 0 )
    ncs += 1
    {INV ∧ ncs > 0}

wants-enter-pedestrian ()
    {INV}
    pueden-peatones.wait (ncn == 0 ∧ ncs == 0)
    np += 1
    {INV ∧ np > 0}

leave-car-north ()
    {INV ∧ ncn > 0}
    ncn -= 1
    if ncn == 0 :
        pueden-sur . signal()
        pueden-peatones . signal ()

leave-car-south ()
    {INV ∧ ncs > 0}
    ncs -= 1
    if ncs == 0 :
        pueden-norte . signal()
        pueden-peatones . signal ()
```

```
leave_pedestrian ()
    {INV ∧ np > 0}
    np -= 1
    if np == 0 :
        peatn_norte .signal ()
        peden_sur .signal ().
```

```
coche_norte :
    wants_enter_north()
    car_north_passing ()
    leave_car_north()

coche_sur :
    wants_enter_south ()
    car_south_passing ()
    leave_car_south ()

peaton :
    wants_enter_pedestrian ()
    pedestrian_passing ()
    leave_pedestrian ().
```

Como el invariante nos justifica que si
$$\begin{cases} ncn > 0 \Rightarrow ncr == 0 \wedge np == 0 \\ ncr > 0 \Rightarrow ncn == 0 \wedge np == 0 \\ np > 0 \Rightarrow ncn == 0 \wedge ncr == 0 \end{cases}$$
, justifica la

seguridad del programa.

Parte 2: Inanición:

```
Monitor_C_P:
  ncn : int = 0
  ncs : int = 0
  np : int = 0
  wn : int = 0  : nº coches norte esperando
  ws : int = 0  : nº coches sur esperando
  wp : int = 0  : nº peatones esperando
  turn : int = -1 : 0: norte, 1: sur, 2: peatones, -1: cualquiera
  puede_norte : VC = True
  puede_sur : VC = True
  puede_peatones : VC = True
  INV : { ncn ≥ 0, ncs ≥ 0, np ≥ 0, ncn·ncs = 0, ncn·np = 0, ncs·np = 0, wn ≥ 0, ws ≥ 0, wp ≥ 0 }


wants_enter_north()
    {INV}
    wn += 1
    puede_norte.wait (ncs == 0 ∧ np == 0 ∧ (turn == -1 ∨ turn == 0))
    turn == 0
    ncn += 1
    wn -= 1
    {INV ∧ ncn > 0}

wants_enter_south()
    {INV}
    ws += 1
    puede_sur.wait (ncn == 0 ∧ np == 0 ∧ (turn == -1 ∨ turn == 1))
    turn == 1
    ncs += 1
    ws -= 1
    {INV ∧ ncs > 0}

wants_enter_pedestrian()
    {INV}
    wp += 1
    puede_peatones.wait (ncn == 0 ∧ ncs == 0 ∧ (turn == -1 ∨ turn == 2))
    turn == 2
    np += 1
    wp -= 1
    {INV ∧ np > 0}
```

```
leave_car_north()
    {INV ∧ ncn > 0}
    ncn -= 1
    turn = -1
    if ws > 5:
        turn = 1
        pueden_sur.signal()
     if wp > 5:
        turn = 2.
        pueden_peatones.signal()
      if ncn == 0:
        pueden_sur.signal()
        pueden_peatones.signal()

leave_car_south()
    {INV ∧ ncs > 0}
    ncs -= 1
    turn = -1
     if wp > 5:
        turn = 2
        pueden_peatones.signal()
     if wn > 5:
        turn = 0
        pueden_norte.signal()
     if ncs == 0:
        pueden_peatones.signal()
        pueden_norte.signal().

leave_pedestrian()
    {INV ∧ np > 0}
    np -= 1
    turn = -1
    if wn > 5:
        turn = 0
        pueden_norte.signal()
     if ws > 5:
        turn = 1
        pueden_sur.signal()
     if np == 0:
        pueden_norte.signal()
        pueden_sur.signal()
```

La seguridad del problema se demuestra de manera análoga al ejercicio anterior.

Veamos la ausencia de deadlock, fijándonos en las funciones leave_car_north, leave_car_south, leave_pedestrian:

Cuando un coche o peatón sale del puente, siempre pone el turno neutro -1 para que en caso de que se active cualquier variable condición, todos puedan tener permiso. Además, en caso de que no salen coches en el puente, se notifica a las tres variables condición, que, con el turno neutro, pueden activarse indistintamente según las hipótesis de justicia.

Finalmente, para demostrar la ausencia de inanición del problema, hacemos uso de los turnos. En caso de que haya más de 5 integrantes de un grupo esperando a entrar, en vez de dejarse el turno neutro, se toma que puedan entrar todos, se fuerza a que ese grupo en concreto entre al puente. Además, en caso de haber varios grupos con mucha gente esperando, se ha configurado un "pase de turno" de un grupo a otro para evitar que el turno solo cambie entre dos grupos. De esta manera aseguramos que no va a haber un único grupo que pase todo el rato o que solo pasen dos y que todos tengan un tiempo máximo de espera.

$$\begin{tikzcd}
& C.S \arrow[dl, bend left, "2"] \arrow[dr, bend left, "1"] & \\
C.N \arrow[ur, bend left, "1"] \arrow[rr, bend left, "2"] & & P \arrow[ul, bend left, "2"] \arrow[ll, bend left, "1"]
\end{tikzcd}$$

(pase de turno en caso de que todos esperen).