```
/*
Resolver con PASAJE DE MENSAJES ASINCRÓNICOS (PMA) el siguiente problema. Se debe simular la
atención en un peaje con 7 cabinas para atender a N vehículos (algunos de ellos son ambulancias).
Cuando el vehículo llega al peaje se dirige a la cabina con menos vehículos esperando y se queda
ahí hasta que lo terminan de atender y le dan el ticket de pago. Las cabinas atienden a los
vehículos que van a ella de acuerdo al orden de llegada pero dando prioridad a las ambulancias;
cuando terminan de atender a un vehículo le dan el ticket de pago.
Nota: maximizar la concurrencia.
*/


//Solución bien, acorde al enunciado

chan pedirCabina(int), darCabina[N](int), pasarPrioridad[7](int), pasarNormal[7]
(int), recibirTicket[7](Ticket), confirmarRecepcion[7](), meVoy(int),
hayVehiculo[7]();

process vehiculo[i: 0..N-1] {
    int idC;

    send pedirCabina(id);
    receive darCabina[id](idC);
    if (Tipo() == "Ambulancia") send pasarPrioridad[idC](id);
    else send pasarNormal[idC](id);
    receive recibirTicket[idC](ticket);
    send confirmarRecepcion[idC]();
    send meVoy(idC);
}

procees cabina[id: 0..6] {
    int idV;

    while (true) {
        receive hayVehiculo[id]();
        Ticket ticket = new Ticket();

        if (empty(pasarPrioridad)) receive pasarNormal[id](idV);
        else receive pasarPrioridad[id](idV);

        send recibirTicket[id](ticket);
        receive confirmarRecepcion[id]();
    }
}

process Coordinador {
    int idV, idC;

    esperaCabina[7] = ([7], 0);

    while (true) {
        if (empty(meVoy)) {
            receive pedirCabina(idV);
            idC = Minimo(esperaCabina);
            send darCabina[idV](idC);
            esperaCabina[idC]++;
            send hayVehiculo[idC]();
        }
        else {
            receive meVoy(idC);
```

```
            esperaCabina[idC]--;
        }
    }
}


// Solución mala utilizando una gran cola general. No resuelve el problema pero
para repasar
// algo parecido al ejercicio 4 de PMA sirvió.

chan EncolarVehiculo(int, String), Atencion[N](int), RecibirTicket[7](Ticket),
ConfirmarRecepcion[7](), PedidoTrabajo(int), Pedido();

process Vehiculo[id: 0..N-1] {
    int idC;
    Ticket ticket;
    String tipo = getTipo();

    send EncolarVehiculo(id, tipo);
    send Pedido();
    receive Atencion[id](idC);
    receive RecibirTicket[idC](ticket);
    send ConfirmarRecepcion[idC]();
}

process Cabina[id: 0..6] {
    int idV;
    while (true) {
        Ticket ticket = new Ticket();

        send PedidoTrabajo(id);
        send Pedido();
        send RecibirTicket[id](ticket);
        receive ConfirmarRecepcion[id]();
    }
}

process Coordinador {
    queue colaVehiculos, colaAmbulancias;
    int idC, idV;

    while (true) {
        receive Pedido();
        if (empty(PedidoTrabajo())) {
            receive EncolarVehiculo(idV, tipo)
            if (tipo == "Ambulancia") push(colaAmbulancias, idV);
            else push(colaVehiculos, idV);
        }
        else {
            if (!empty(colaAmbulancias)) pop(colaAmbulancias, idV);
            else if (!empty(colaVehiculos)) pop(colaVehiculos, idV);
            else {
                receive EncolarVehiculo(idV, tipo);
                receive Pedido();
            }
            send Atencion[idV](idC);
        }
    }
}
```