Monitor Bridge

Var
currentWeight : integer; // total weight of cars currently on bridge
maxWeight : integer; // W, global weight limit
freeLanes : integer; // number of free lanes (0..2)
waitingCount : integer; // number of cars waiting in FIFO
queue : condition; // FIFO queue of waiting cars

Procedure InitBridge(W : integer)
Begin
maxWeight := W;
currentWeight := 0;
freeLanes := 2; // two lanes initially free
waitingCount := 0;
End;

Procedure RequestToCross(carWeight : integer)
Begin
// Car arrives and waits in FIFO if it cannot be admitted now
waitingCount := waitingCount + 1;

While ((freeLanes = 0) OR (currentWeight + carWeight > maxWeight)
OR (queue not head of this car)) do
queue.wait; // wait in arrival order

// This car is now at head of queue AND constraints satisfied
waitingCount := waitingCount - 1;
freeLanes := freeLanes - 1;
currentWeight := currentWeight + carWeight;
End;

Procedure LeaveBridge(carWeight : integer)
Begin
// Car leaves: free its lane and weight
freeLanes := freeLanes + 1;
currentWeight := currentWeight - carWeight;

// Wake up next waiting car (FIFO discipline ensured by condition queue)
If waitingCount > 0 then
queue.signal;
End;

End; // end Monitor Bridge

Process P_Car_i(weight_i : integer)
Begin
Call Bridge.RequestToCross(weight_i);
CrossBridge(); // critical section: uses 1 lane
Call Bridge.LeaveBridge(weight_i);
End.

1.