Peterson's Mutual-Exclusion Algorithm (broken version)

```
MODULE agent(req_self,req_other,turn,num)
  VAR
    pc : {L0, L1, L3, L4 };

  ASSIGN
    init(req_self):= 0;
    init(pc):= L0;

    next(pc):=
      case
        pc = L0          : { L0, L1 };
        pc = L1                  : L3;
        pc = L3 & ( req_other ) : L3;
        pc = L3                  : L4;
        pc = L4                  : L0;
     esac;

    next(req_self):=
      case
        pc = L1 : 1;
        pc = L4 : 0;
        1       : req_self;
      esac;

  DEFINE
    request:=  pc = L1;
    critical:= pc = L4;

  FAIRNESS
    running
```

agent0:

L0:	Idle;
L1:	req0 = 1; //request to enter critical section
L3:	while (req1 == 1);
L4:	req0 = 0; //clear the request

agent1:

L0:	Idle;
L1:	req1 = 1; //request to enter critical section
L3:	while (req0 == 1);
L4:	req1 = 0; //clear the request

```
MODULE main
  VAR
    turn  : boolean;
    req0  : boolean;
    req1  : boolean;

    agent0 : process agent(req0,req1,turn,0);
    agent1 : process agent(req1,req0,turn,1);

    safe: SPEC AG( !agent0.critical | !agent1.critical );
    fairness0: SPEC AG( agent0.request -> EF agent0.critical );
    fairness0: SPEC AG( agent1.request -> EF agent1.critical );
```


Peterson's Mutex (Fixed)

```
MODULE agent(req_self,req_other,turn,num)
VAR
  pc : {L0, L1, L2, L3, L4 };
ASSIGN
  init(req_self):= 0;
  init(pc):= L0;

  next(pc):=
    case
      pc = L0                      : { L0, L1 };
      pc = L1                             : L2;
      pc = L2                             : L3;
      pc = L3 & (turn = !num) & req_other: L3;
      pc = L3                             : L4;
      pc = L4                             : L0;
   esac;

  next(req_self):=
    case
      pc = L1 : 1;
      pc = L4 : 0;
      1       : req_self;
   esac;

  next(turn):=
    case
      pc = L2 : !num;
      1       : turn;
   esac;

DEFINE
  request:=  pc = L1;
  critical:= pc = L4;
FAIRNESS
  running
```

agent0:

L0:	Idle;
L1:	req0 = 1; //request to enter critical section
L2:	turn = 1; //give priority to the other procress
L3:	while (turn == 1 && req1 == 1);
L4:	req0 = 0; //clear the request

agent1:

L0:	Idle;
L1:	req1 = 1; //request to enter critical section
L2:	turn = 0; //give priority to the other procress
L3:	while (turn == 0 && req0 == 1);
L4:	req1 = 0; //clear the request

```
MODULE main
  VAR
    turn  : boolean;
    req0  : boolean;
    req1  : boolean;

    agent0 : process agent(req0,req1,turn,0);
    agent1 : process agent(req1,req0,turn,1);

    safe: SPEC AG( !agent0.critical | !agent1.critical );
    fairness0: SPEC AG( agent0.request -> EF agent0.critical );
    fairness0: SPEC AG( agent1.request -> EF agent1.critical );
```