Esercizio di sincronizzazione tra Processi: lo Stadio

Testo:

Errore. Non si possono creare oggetti dalla modifica di codici di campo.

Uno stadio ospita un torneo internazionale di calcio al quale partecipano squadre **italiane** ed **non** (quindi straniere). L'orario delle partite del torneo prevede il continuo susseguirsi di incontri durante l'intero arco di ogni giornata. Si prevede quindi una continua affluenza di tifosi in ingresso ed in uscita dallo stadio.
Lo stadio e` accessibile attraverso due corridoi (***Cancello Nord*** e ***Cancello Sud***) utilizzabili sia per l'**ingresso** che per l'**uscita**.
I tifosi accedono allo stadio in **gruppi** (ognuno dei quali deve essere considerato indivisibile nell'accesso e nell'uscita): a gruppi di **consistenza numerica maggiore** deve essere associata una **maggiore priorità** nell'accesso e nell'uscita dallo stadio.
Per motivi di sicurezza si e` deciso di gestire ogni corridoio in modo tale che gruppi di tifosi di squadre italiane ***non possano incrociarsi*** (nello stesso corridoio) con gruppi di tifosi di squadre straniere: quindi, ogni corridoio può funzionare a doppio senso di percorrenza soltanto nel caso in cui gli utenti che lo attraversano siano tutti tifosi dello stesso tipo (cioè tutti italiani, oppure tutti stranieri). A questo proposito, si assuma che tutti i gruppi siano omogenei (o italiani o stranieri).
Per ragioni di ospitalità, inoltre, si e` stabilito di **favorire** *ragionevolmente* (nell'accesso e nell'uscita dallo stadio) **i gruppi di tifosi stranieri**.
Lo stadio ha una capienza massima **MAX_C** (che esprime il numero massimo di tifosi consentito all'interno dello stadio) oltre la quale non è permesso l'accesso di ulteriori persone.
Infine, nel tentativo di evitare situazioni di saturazione, tifosi che desiderano uscire dallo stadio devono avere la precedenza sui tifosi che intendono entrare.
Si definisca una politica di gestione dello stadio che risponda alle specifiche date. Descrivere la politica e la si implementi usando il costrutto monitor.

Soluzione:

```
program Esame;
const   MAX_C=...;   {Capacita` massima dello stadio}
        MAX_G=...;   {Consistenza massima per i gruppi di tifosi: MAX_G<MAX_C}

type    direzione=(in,out);                 {verso di percorrenza per i cancelli}
        nazionalita=(italiani,stranieri);   {tipo dei gruppi}
        cancello=(nord, sud);
        consistenza=1..MAX_G;

type    gestore_stadio=monitor;
         var    tifosi_in_stadio:0..MAX_C;
                n_corridoio: array[cancello, nazionalita, direzione] of  0..MAX_C;
                coda_in: array[cancello, nazionalita, consistenza] of condition;
                coda_out: array[cancello, nazionalita, consistenza] of condition;

function otherN(t:nazionalita): nazionalita;
begin
        if t = italiani    then otherN:=stranieri
                           else otherN:=italiani;
end;

function otherC(c:cancello): cancello;
begin
if C = nord      then otherC:=sud else otherC:=nord;
end;

function cecoda_out(c: cancello; t: nazionalita);
var     i: consistenza;
        flag: boolean;
begin
        flag:=false;
        for i :=MAX_G downto 1
                do   begin
                        if coda_out[c,t,i].queue then flag:=true;
                     end;
        cecoda_out:=flag;
end;

procedure segnala_in(c: cancello; tipo: nazionalita)
var i: consistenza;
begin
        for  i: = MAX_C - n_tifosi   downto  1
                do     coda_in[c, tipo, i].signal;
end;
```

```
procedure segnala_out(c: cancello; tipo: nazionalita)
var i: consistenza;
begin
        for  i := MAX_C - n_tifosi   downto  1
                do
                    coda_out[c, tipo, i].signal;
end;

procedure entry acq_in(c: cancello;  tipo: nazionalita; num: consistenza);
begin
        while   ((tifosi_in_stadio+num>MAX_C) or          {non c'e` posto nello stadio}
                 (n_corridoio[c,otherN(tipo), out]>0) or      {ci sono gruppi di tipo opposto in dir opposta}
                 (cecoda_out(c, otherN(tipo)) ))               {ci sono gruppi di tipo opposto in coda per uscire}
                        do  coda_in[c,tipo,num].wait;

        n_corridoio[c, tipo, in]:=n_corridoio[c, tipo, in]+num;
        tifosi_in_stadio:=tifosi_in_stadio+num;
        coda_in[c, tipo, num].signal;          {segnalazione al processo omologo}
end;

procedure entry ril_in(c_in: cancello;  tipo: nazionalita; num:1..MAX_G);
begin
        n_corridoio[c,tipo, in]:=n_corridoio[c,tipo, in] - num;
        segnala_out(c, otherN(tipo));
end;

procedure entry acq_out(c_out: cancello; tipo: nazionalita; num: 1..MAX_G);
begin
        while   (n_corridoio[c,otherN(tipo),in]>0) or {ci sono gruppi di tipo opposto in dir. opposta}
                        do  coda_out[c,tipo,num].wait;

        n_corridoio[c, tipo, out]:=n_corridoio[c, tipo, out]+num;
        tifosi_in_stadio:=tifosi_in_stadio-num;    {rilascio di capacita`}

        segnala_in(c, tipo);              {segnalazioni in seguito al rilascio di capacita` nello stadio}
        segnala_in(otherC(c), stranieri);
        segnala_in(otherC(c), italiani);
        coda_out[c, tipo, num].signal;          {segnalazione al processo omologo}
end;

procedure entry ril_out(c_out: cancello;  tipo: nazionalita; num: 1..MAX_G);
begin
        n_corridoio[c,tipo, out]:=n_corridoio[c,tipo, out] - num;
        segnala_in(c, otherN(tipo));
end;
```

```
begin   {inizializzazione variabili del monitor}
    tifosi_in_stadio:=0;
    n_corridoio[nord, stranieri, in]: = 0;
    n_corridoio[nord, stranieri, out]: = 0;
    n_corridoio[nord, italiani, in]: = 0;
    n_corridoio[nord, italiani, out]: = 0;
    n_corridoio[sud, stranieri, in]: = 0;
    n_corridoio[sud, stranieri, in]: = 0;
    n_corridoio[sud, italiani, in]: = 0;
    n_corridoio[sud, italiani, out]: = 0;
end;    {fine monitor}

type gruppo=process(num: 1..MAX_G; tipo: nazionalita);
var     c_in, c_out: cancello;
begin
        <decisione cancello ingresso-->c_in>
        Stadio.acq_in(c_in, num, tipo);
        Stadio.ril_in(c_in, num, tipo);
        <permanenza nello stadio e decisione cancello uscita --> c_out>
        Stadio.acq_out(c_out, num);
        Stadio.ril_out(c_out, num, tipo);
end;

var     Stadio: gestore_stadio;
        g1(....), g2(...), .., gn(..): gruppo;

begin   {applicazione concorrente}
end;
```