Si programmi un sistema in cui un insieme di processi condivide un array A di 3 interi (indici 0, 1 e 2).
I valori iniziali dell'array sono tutti 0 (A[0]=A[1]=A[2] = 0).
Esistono le seguenti classi di processi:

- processi scrittori: iterativamente selezionano un indice 0 <= i <= 2 ed un valore intero v ed assegnano v ad A[i].
- processi lettori: iterativamente calcolano A[0]+A[1]+A[2].

Condizioni:

- le race condition devono essere impossibili;

- quando uno o piu' lettori stanno lavorando sull'array ed uno scrittore va in waiting perche' tenta di accedere all'array (va in waiting perche' altrimenti potrebbe verificarsi una race condition), da questo momento in poi al piu' 10 lettori che tentano di accedere all'array possono accedere prima che almeno uno scrittore in attesa venga risvegliato. (Questa condizione garantisce che gli scrittori non
rimangano in attesa all'infinito "per colpa" dei lettori).

- quando uno o piu' scrittori stanno lavorando sull'array (ovviamente in posizioni diverse se si tratta di piu' scrittori) ed un lettore va in waiting perche' tenta di accedere all'array (va in waiting perche' altrimenti potrebbe verificarsi una race condition), da questo momento in poi al piu' 4 scrittori che tentano di accedere ad A[0], al piu' 4 scrittori che tentano di accedere ad A[1] ed al piu' 4 scrittori che tentano di accedere ad A[2] possono accedere prima che almeno un lettore venga risvegliato.
(Questa condizione garantisce che i lettori non rimangano in attesa all'infinito "per colpa" degli scrittori).

SEMAFORI:

mutex: semaforo per ME
sem_R: semaforo per i lettori;
sem_i: semaforo per gli scrittori su A[i];

VARIABILI

wait_R = numero lettori in waiting.
wait_i = numero scrittori in waiting su A[i];
run_R = numero lettori working;
run_i = true IFF uno scrittore sta lavorando su A[i];
x = bonus lettori;
x_i = bonus scrittori su i.

READER:

```
while(true){
   wait(mutex);
   while( (run_0 OR run_1 OR run_2) OR
          ((wait_0>0 OR wait_1>0 OR wait_2>0) & x==0)
        ){
     wait_R = wait_R + 1;
     signal(mutex);
     wait(sem_R);
     wait(mutex);
   }
   run_R = run_R+1;
   (*) x_0=x_1=x_2= 4;
   if(wait_0>0 OR wait_1>0 OR wait_2>0){
     x = x - 1;
   }
   signal(mutex);

   int Z = A[0]+A[1]+A[2];

   wait(mutex);
   run_R = run_R - 1;
   if(run_R==0){
     (**) x = 10;
     for(int i=0; i<3; i=i+1;){
        if(wait_i >0){
          wait_i = wait_i - 1;
          signal(sem_i);
        }
     }
   }
   signal(mutex);
}
```

WRITER:

```
while(true){
  int i = ....;  int v = ....;
  wait(mutex);
  while( run_R > 0 OR run_i OR
         (wait_R > 0 & x_i = 0)
       ){
     wait_i = wait_i + 1;
     signal(mutex);
     wait(sem_i);
     wait(mutex);
```

```
  }
  run_i = true;
  (*) x = 10;
  if(wait_R>0){
    x_i = x_i - 1;
  }
  signal(mutex);
  A[i] = v;
  wait(mutex);
  run_i = false;
  if(x_i > 0 & wait_i>0){
    wait_i=wait_i-1;
    signal(sem_i);
  }
  if(!run_((i+1)%3) & !run_((i+2)%3 & !run_i){
    (**) x_0=x_1=x_2 = 4;
    while(wait_R >0){
      wait_R = wait_R -1;
      signal(sem_R);
    }
  }
  signal(mutex);


}
```