Si consideri il classico problema dei produttori e consumatori, con il buffer implementato con un array di interi di dimensione 100. Si assuma, per semplicita', che gli interi prodotti e consumati siano >=0. Si considerino le seguenti condizioni aggiuntive:

1.Gli interi pari possono occupare solo le posizioni da 0 a 49;

2.Gli interi dispari possono occupare solo le posizioni da 50 a 99;

3.Vi e' una nuova categoria di processi, i processi ConsPari, che consumano tutti gli interi pari, se l'array contiene almeno 20 interi pari;

Quanto un processo tenta di effettuare un'operazione al momento non consentita (per esempio produrre un intero che il buffer non può al momento ospitare), il processo deve essere messo in attesa. Programmare il sistema sfruttando i semafori con la semantica tradizionale.

variabili:
attesaProd=0; // numero di produttori in attesa
attesaCons=0; // numero di consumatori in attesa
attesaCons20=0; // numero di consumatori ConsPari in attesa
numPari = 0; // numeri pari presenti nell'array.

semafori:
semCons=0; // semaforo per mettere in attesa i consumatori
semProd=0; // semaforo per mettere in attesa i produttori
semCons20=0; // semaforo per mettere in attesa i ConsPari
mutexPari=1; // semaforo per garantire m.e. sulle variabili condivise.

NB: eccetto mutexPari, questi semafori non diventeranno mai > 0.

```
Producer(){
 while(true){
 .....
 int item = ...;
 if(item%2==0){

    wait(mutexPari){
    if(numPari==50){
      attesaProd++;
      signal(mutexPari);
      wait(semProd);
      wait(mutexPari);
    }
```

```
    buffer[i]=item;
    i=(i+1)%50;
    numPari++;
    if(attesaCons>0){
      attesaCons--;
      signal(semCons);
    }
    if(numPari==20 & attesaCons20>0){
      attesaCons20--;
      signal(semCons20);
    }
    signal(mutexPari);

 }
 else{
   facile! //
 }
 .....
 }
}
```

```
Consumer(boolean pari){
 while(true){
   ....
   if(pari){

     wait(mutexPari){
     while(numPari==0){
       attesaCons++;
       signal(mutexPari);
       wait(semCons);
       wait(mutexPari);
     }
     int item=buffer[j];
     j=(j+1)%50;
     numPari--;
     if(attesaProd>0){
       attesaProd--;
       signal(semProd);
     }

     signal(mutexPari);
   }
   else{
    facile!
   }
   ....
```

```
 }
}



Consumer20(){
 while(true){
   ....


     wait(mutexPari){
     while(numPari<20){
       attesaCons20++;
       signal(mutexPari);
       wait(semCons20);
       wait(mutexPari);
     }
     consumaTutto();
     j=0;numPari = 0;
     int k=0;
     while(attesaProd>0 & k<50){
       attesaProd--;
       signal(semProd);
       k++;
     }
     signal(mutexPari);

   ....
 }
}
```