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 tutti strettamente positivi oppure uguali a 0, vale a dire >=0.
Si considerino le seguenti condizioni aggiuntive:

1. Gli interi pari possono occupare solo le posizioni di indice pari (0, 2, 4, …., 98);

2. Gli interi dispari divisibili per 3 possono occupare solo le posizioni di indice dispari divisibile per 3 (3, 9, 15, 21,..., 99).

3. Gli interi dispari che non sono divisibili per 3 possono occupare solo le posizioni di indice dispari non divisibili per 3 (1, 5, 7, 11, 13, 17.., 97).

Quando un processo tenta di effettuare un'operazione al momento non consentita (per esempio produrre un intero che il buffer non puo' al momento ospitare), il processo deve essere sospeso.

Programmare il sistema sfruttando i semafori con la semantica tradizionale

Semafori:

emptyP = 50 // n. di posizioni libere che possono ospitare numeri pari
empty3 = 17 // n. di posizioni libere che possono ospitare numeri dispari div. per 3
emtpyD = 33 // n. di posizioni libere che possono ospitare numeri dispari non div. per 3
mutexP = 1 // m.e. su zone dell'array che ospitano pari
mutex3 = 1 // m.e. su zone dell'array che ospitano dispari div. per 3
mutexD = 1 // m.e. su zone dell'array che ospitano dispari non div. per 3

Si assume l’array inizializzato con interi NEGATIVI.

produttore:

```
while(true){
 ......
 item=....;

 if(item%2==0){
   wait(emptyP);
   wait(mutexP);
   buffer[i]=item;
   i=(i+2)%100;
   signal(mutexP);
 }

 if(item%2!=0 & item%3==0){
   wait(empty3);
   wait(mutex3);
   buffer[j]=item;
   if(j==99){j=3;}else{j=j+6;}
   signal(mutex3);
 }

 if(item%2!=0 & item%3!=0){
   wait(emptyD);
   wait(mutexD);
   buffer[k]=item;
   k++;
   while(k%2==0 | k%3==0){k=(k+1)%100;}
   signal(mutexD);
 }

 signal(full);

}
```

consumatore:

```
while(true){
 wait(full);
 wait(mutexP);
 wait(mutexD);
 wait(mutex3);
 while(buffer[h]<0){h++;}
```

```
 item=buffer[h];
 buffer[h]=-1;
 h=h+1%100;
 if(item%2==0){signal(emtpyP);}
 if(item%2!=0 & item%3==0){signal(empty3);}
 if(item%2!=0 & item%3!=0){signal(emptyD);}
 signal(mutexD);
 signal(mutex3);
 signal(mutexP);

}
```