Università degli Studi di Napoli Federico II
Corso di Laurea in Ingegneria Informatica
Esame di Sistemi Operativi
Proff. Cinque, Cotroneo, Natella

Prova pratica del 21/12/2015
Durata della prova: 150 minuti

Cognome ....……................. Nome ...….............................. Matr. ....

°°°

Lo studente legga attentamente il testo e produca il programma, il makefile, ed i casi di test necessari per dimostrarne il funzionamento. La mancata compilazione dell'elaborato, la compilazione con errori o l'esecuzione errata del programma daranno luogo alla valutazione come ***prova non superata****. Ricordarsi di indicare Nome, Cognome e matricola su questo stesso foglio, che dovrà essere in ogni caso consegnato alla Commissione. Al termine della prova lo studente dovrà fare verificare il funzionamento del programma ad un membro della Commissione.*

°°°

Testo della prova

Si realizzi in linguaggio C/C++ un'applicazione **multithread** utilizzando il costrutto **Monitor**, che simuli la gestione di un vettore di buffer di I/O. Per gestire l'accesso al vettore, si utilizzi la soluzione del **produttore/consumatore con vettore circolare**.

```
typedef struct {
      int indirizzo;
      int dato;
} Buffer;
typedef struct {
      Buffer vettore[10];
      // ... inserire qui variabili aggiuntive per la sincronizzazione ...
} GestioneIO;
void Inizializza(Gestione IO * g);
void Produci(GestioneIO * g, Buffer * b);
int Consuma(GestioneIO * g, Buffer * b);
```

Il metodo *Inizializza* imposta il valore iniziale delle variabili interne del monitor. Il metodo *Produci* ha un parametro di ingresso di tipo puntatore a *Buffer*, per il passaggio di un buffer da copiare nel vettore. Il metodo *Produci* **deve sospendere il thread chiamante se non vi è spazio disponibile**.

Il metodo *Consuma* preleva dal vettore il contenuto di un buffer, che deve essere dato al chiamante attraverso il puntatore a *Buffer* passato come parametro di ingresso. Nel caso non vi siano buffer disponibili, il metodo *Consuma* **non deve sospendere il thread chiamante**, bensì deve indicare al chiamante l'assenza di buffer pieni, ritornando il valore 1 attraverso il parametro di uscita intero della funzione. Nel caso invece che vi sia un buffer disponibile da consumare, la funzione dovrà ritornare 0.

Il programma principale dovrà creare 4 thread Produttori e 2 thread Consumatori. I thread *Produttori* effettuano ciascuno 3 produzioni, attendendo 1 secondo tra una produzione e la successiva (usando la primitiva *sleep*). I produttori dovranno scegliere a caso (tra 0 e 10) i valori di *indirizzo* e *dato* da usare alla prima produzione, e incrementarli di 1 ad ogni produzione. Ad esempio, il produttore può scegliere la coppia (3,7) alla prima produzione, e usare le coppie (4,8) e (5,9) alle produzioni successive.

Ogni thread Consumatore dovrà effettuare 4 iterazioni. Ad ogni iterazione, il Consumatore effettua **consumazioni ripetute** (chiamando più volte il metodo *Consuma*, e stampando a video i valori prelevati), fino a quando **il vettore di buffer diventa vuoto** (il valore di ritorno di *Consuma* è pari a 1). Il thread Consumatore deve attendere 3 secondi (tramite *sleep*) tra due iterazioni.