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

## Prova pratica del 24/11/2016 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**, basata sul costrutto **Monitor**, che realizzi lo schema **produttore/consumatore con coda circolare**.

```
typedef struct {
    int vettore[5];
    int testa, coda;
    int produttori_in_attesa;
    // ... inserire qui le variabili per la sincronizzazione ...
} MonitorCoda;
// il valore di ritorno di produzione() indica se il monitor è sovraccarico
int produzione(MonitorCoda *m, int valore);
// il valore di ritorno di consumazione() è il valore prelevato dalla coda
int consumazione(MonitorCoda *m);
```

In aggiunta allo schema di sincronizzazione studiato al corso, si richiede la seguente variante. Si supponga di voler impedire che un numero eccessivo di thread produttori si accumuli in attesa all'interno del monitor, per evitare una condizione di sovraccarico. I thread produttori, nel caso in cui il vettore di buffer sia pieno, dovranno quindi verificare se il numero di thread già in attesa è minore di 3 (usando la variabile produttori\_in\_attesa). Se i thread in attesa sono meno di 3, il thread produttore dovrà sospendersi in attesa che un buffer si renda disponibile. Invece, se vi sono già 3 thread in attesa, il monitor dovrà forzare l'uscita del thread produttore, facendo return di un valore diverso da 0.

Il thread chiamante del metodo produzione(), nel caso sia ritornato un valore diverso da 0, dovrà attendere 3 secondi e riprovare la produzione, fin quando la produzione non abbia successo.

```
for (4 produzioni) {
    while(1) {
        ret = produzione(monitor, valore);
        if(ret == 0) { break; }
        sleep(3);
    }
    sleep(1);
}
```

Per i thread consumatori, non è richiesto alcun controllo sul numero di thread già in attesa. Il programma principale dovrà creare 4 produttori e 1 consumatore. Ogni produttore dovrà produrre 4 valori, e il consumatore consumarne 16. I thread produttori dovranno attendere 1 secondo tra le produzioni, e i consumatori dovranno attendere 2 secondi tra le consumazioni.