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

## Prova pratica 18/04/2023 Durata della prova: 75 minuti

Lo studente completi il programma a corredo di questo documento, in base alle indicazioni qui riportate. La prova sarà valutata come segue:

- A: Prova svolta correttamente.
- **B**: Il programma non esegue correttamente, con errori minori di programmazione o di concorrenza.
- C: Il programma non esegue correttamente, con errori significativi (voto max: 22).
- INSUFFICIENTE: Il programma non compila o non esegue, con errori gravi di sincronizzazione.

## Testo della prova

Si realizzi in linguaggio C/C++ un programma multiprocesso, basato sul costrutto Monitor (di tipo "signal and continue"), che realizzi lo schema del produttore-consumatore con vettore di stato. È previsto che i produttori possano inserire elementi di due tipi differenti. I buffer del vettore possono essere in 4 possibili stati: LIBERO, IN\_USO, OCCUPATO1, OCCUPATO2 (questi ultimi due stati a seconda del tipo di elemento che è stato prodotto). I produttori chiamano uno fra due differenti metodi ("produci\_tipo\_1" e "produci\_tipo\_2") per effettuare la produzione sui due tipi differenti. Analogamente, i consumatori chiamano uno fra due metodi differenti ("consuma\_tipo\_1" e "consuma\_tipo\_2") per consumare.

Per verificare il funzionamento del programma, si creino 2 processi produttori, che effettuino 4 produzioni ciascuno di tipo 1; altri 2 processi produttori, che effettuino 4 produzioni ciascuno di tipo 2; 1 processo consumatore, che effettui 8 consumazioni di tipo 1; 1 processo consumatore, che effettui 8 consumazioni di tipo 1. Il vettore di buffer conterrà al più 4 elementi.

```
#define DIM 4

typedef struct {
    int vettore[DIM];
    int stato[DIM];
    int num_liberi;
    int num_occupati_tipo1;
    int num_occupati_tipo2;
    // ... aggiungere ulteriori variabili per la sincronizzazione
} MonitorPC;

void inizializza(MonitorPC * m);
void rimuovi(MonitorPC * m);
void produci_tipo_1(MonitorPC * m, int valore);
void produci_tipo_2(MonitorPC * m, int valore);
void consuma_tipo_1(MonitorPC * m, int & valore);
void consuma_tipo_1(MonitorPC * m, int & valore);
void consuma tipo_2(MonitorPC * m, int & valore);
```