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

Simulazione d'Esame
Durata della prova: 75 minuti

Lo studente completi il programma a corredo di questo documento, seguendo le seguenti indicazioni.

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.

Istruzioni per la consegna dell'elaborato

L'elaborato dovrà essere svolto in una cartella dal nome: `Cognome_Nome_Matricola_Docente`

Esempio:

```
mkdir Rossi_Mario_N46012345_Cotroneo
cd Rossi_Mario_N46012345_Cotroneo
```

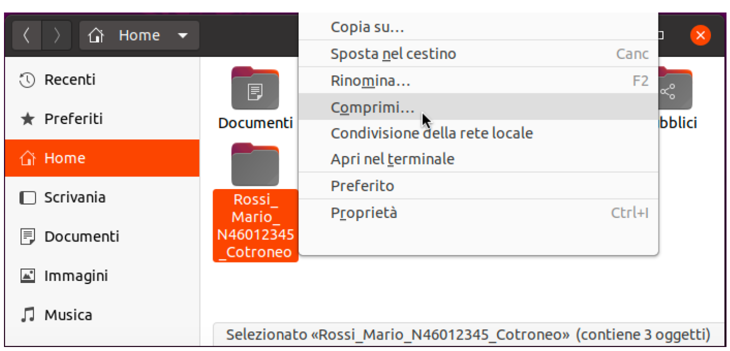
…. copiare nella cartella i file forniti per l'esame ….

…. sviluppare il programma ….

…. per la consegna, dalla shell (assumendo di essere ancora nella cartella di lavoro), creare un file compresso ("tar") con i seguenti comandi:

```
cd ..
tar -czvf ./Rossi_Mario_N46012345_Cotroneo.tar.gz  ./Rossi_Mario_N46012345_Cotroneo
```

In alternativa, è consentito creare il file compresso "tar" tramite l'interfaccia grafica.



All'interno della macchina virtuale, aprire il browser all'indirizzo specificato dal docente e scaricare i file da completare.

Alla consegna, attendere una notifica del docente.

Testo della prova

Si realizzi in linguaggio C/C++ un'applicazione **multiprocesso** e **multithread** (libreria PThread) per parallelizzare dei calcoli matematici, basata su costrutto **monitor** e **code di messaggi**, e sullo **schema produttore-consumatore**.

```
typedef struct {

    int operandi[DIM_BUFFER];
    // TO BE COMPLETED

} MonitorOperandi;
```

```
typedef struct {
    // TO BE COMPLETED
} Risultato;
```

```
void inserisci_operando(MonitorOperandi * mo, int operando);
int *preleva_operando(MonitorOperandi * mo);
void inserisci_risultato(Risultato *m , int queue);
int preleva_risultato(Risultato *m, int queue, int tipomess);
void * genera_operandi(void *);
void * calcola(void *);
void preleva_risultati(int);
```

Il programma dovrà prevedere 2 gruppi di thread e 1 processo. Il primo gruppo (2 thread) dovrà **generare gli operandi**: ogni thread dovrà invocare 6 volte il metodo `inserisci_operando()` e passando dei valori casuali tra 1 e 10. Il valore andrà inserito nel vettore `operandi[]`, e se già pieno, il thread chiamante viene posto in attesa.

Il secondo gruppo (3 thread) dovrà **prelevare gli operandi** chiamando il metodo `preleva_operando()` (2 chiamate per ogni thread), che ritorna **due valori da estrarre** (i.e., un array di 2 elementi) dal vettore `operandi[]`. Se tale vettore non contiene almeno 2 elementi, il thread chiamante viene posto in attesa. La sincronizzazione per l'accesso al buffer operandi deve essere fatta con **coda circolare**.

Dopo aver prelevato i due operandi, i thread del secondo gruppo dovranno **effettuare il calcolo** della somma dei quadrati dei valori prelevati, attendere 1 secondo, ed inserire il risultato nella **coda risultati** chiamando il metodo `inserisci_risultato()`, che invia il risultato ottenuto tramite **coda di messaggi** e *send sincrona*.

Un singolo processo dovrà **prelevare i risultati** dalla coda risultati, chiamando ripetutamente (**6 volte in totale**) il metodo `preleva_risultato()` che riceverà il messaggio dalla coda risultati. Se non c'è nessun messaggio sulla **coda risultati**, il processo chiamante deve bloccarsi. Il processo dovrà contare il numero di volte che il risultato ricevuto è minore o uguale a 25, e il numero di volte che il risultato è maggiore di 25. Al termine di tutti i prelievi, il processo dovrà stampare i due conteggi.