Nome_____ Cognome _____N. di matricola (10 cifre) _____ Riga ____ Col ____ .

UNIVERSITA' DEGLI STUDI DI BOLOGNA - CORSO DI LAUREA IN INFORMATICA
PROVA **PARZIALE** DI SISTEMI OPERATIVI
ANNO ACCADEMICO 2024/2025
23 gennaio 2025

Esercizio -1: Essere iscritti su AlmaEsami per svolgere questa prova.

Esercizio 0: Scrivere correttamente nome, cognome, matricola e posizione in tutti i fogli prima di svolgere ogni altro esercizio. Scrivere esclusivamente a penna senza abrasioni. E' vietato l'uso delle penne cancellabili, della matita, dei coprenti bianchi per la correzione (bianchetto) e la scrittura in colore rosso (riservato alla correzione).
Il compito e' formato da due fogli, quattro facciate compresa questa. Le soluzioni che si vogliono sottoporre per la correzione devono essere scritte negli spazi bianchi di questi fogli. Non verranno corretti altri supporti.
E' obbligatorio consegnare il compito, e' possibile chiedere che esso non venga valutato scrivendo “NON VALUTARE” in modo ben visibile nella prima facciata.
Per svolgere questo compito occorre solo una penna e un documento di identità valido. La consultazione o anche solo la disponibilità di altro materiale comporterà l'annullamento del compito (verrà automaticamente valutato gravemente insufficiente).

Esercizio c.1: Il monitor count bounded buffer (`cbb`) gestisce un buffer limitato di `MAX` elementi.
`cbb` ha due procedure entry:

```
void write(obj_t el)
int remove(obj_t el)
```

La funzione `write` inserisce un elemento nel buffer. Se vi sono già `MAX` elementi i processi che chiamano la funzione write devono attendere in ordine FIFO.
La funzione `remove` deve eliminare dal buffer tutti gli elementi di valore uguale al parametro `el` presenti nel buffer al momento della chiamata. Il valore di ritorno rappresenta il numero di elementi cancellati dal buffer. Se non vi sono elementi restituisce zero.

Esercizio c.2: Dato un servizio di message passing asincrono, implementare (senza far uso di processi server) un servizio semisincrono che fornisce le seguenti funzioni:

```
sssend(msg_t msg, pid_t dest)
msg_t ssrecv(pid_t sender)
```

Se ci sono meno di 7 messaggi inviati dal processo A al processo B tramite la chiamata `sssend` non ancora ricevuti da B tramite la `ssrecv` la chiamata `sssend` non è bloccante. Se ci sono 7 messaggi non ancora ricevuti la chiamata `sssend` è bloccante. Il parametro `sender` della funzione `ssrecv` è l'identificativo di un processo mittente; il caso ANY non è consentito.

Esercizio C.3: Siano dati i due processi seguenti `A` e `B` nei quali `print` è una funzione atomica.

```
semaphore SA=1
semaphore SB=0

process A:
  while true:
    SA.P()
    <print('A')>
    SB.V()

process B:
  while true
    SB.P()
    <print('B')>
    SA.V()
```

Modificare il codice, senza aggiungere ulteriori semafori o chiamate della funzione `print` in modo che l'output dei due processi sia la stringa infinita BBBABBBABBBABBBA.....