COGNOME _____ NOME_____ **LT ◊ LM ◊** MATR. _____

SISTEMI OPERATIVI – Prova scritta

A – In un complesso robotizzato carrelli a controllo autonomo pervengono ad un sistema con elevatori per cambio piano su 3 distinte code (A0..A2): ogni coda ospita carrelli che devono essere instradati nella rispettiva delle 3 tratte di uscita (C0..C2) su altro piano (in pratica, in Ai c'è una coda FIFO di carrelli tutti da instradare su Ci). Tramite il convogliatore I1, che può spostare un solo carrello alla volta, un carrello può essere movimentato in uno qualsiasi dei due elevatori di transito (B0, B1), posto che sia quell'elevatore che la corrispondente tratta di uscita siano liberi; se questa condizione non è soddisfatta, il carrello rimane in attesa nella coda di ingresso (e quindi anche tutti quelli che lo seguono in coda). La movimentazione dall'elevatore Bi a una tratta Ck avviene mediante il meccanismo I2, impegnabile, come I1, con un solo carrello alla volta. Sia gli elevatori Bi che le tratte Ck , quindi, sono intese a capienza unitaria (vedi figura).
Utilizzando una singola Regione come unico strumento di sincronizzazione, si simuli il controllo del sistema di cambio piano con la classe Smista che dispone dei seguenti metodi (chiamati dai thread che rappresentano i carrelli autonomi):

- **`int instrada(final int i)`**

viene chiamato da un carrello che si dispone nella coda di attesa A<i> per l'instradamento in C<i>; dal metodo si ritorna quando il carrello ha ottenuto il consenso a essere prelevato da I1; il metodo restituisce l'indice (h=0 o 1) dell'elevatore Bh selezionato che risulta così impegnato per il carrello richiedente;

- **`void outI1 (int h)`**

il carrello si trasferisce nell'elevatore di transito indicato da **`h`** e lascia libero il movimentatore I1;

- **`int outB (int h)`**

il carrello attende la disponibilità del movimentatore I2; dal metodo si ritorna quando il carrello ha ottenuto il consenso a essere prelevato da I2, lasciando libero l'elevatore B<h>; il metodo restituisce l'indice (i=0, 1 o 2) della tratta finale C<i> associata al carrello;

- **`void inC (int i)`**

il carrello lascia il movimentatore I2 liberandolo ed entra nella tratta C<i>;

- **`void outC (int i)`**

il carrello lascia definitivamente il sistema con elevatori, liberando la tratta d'uscita C<i>.

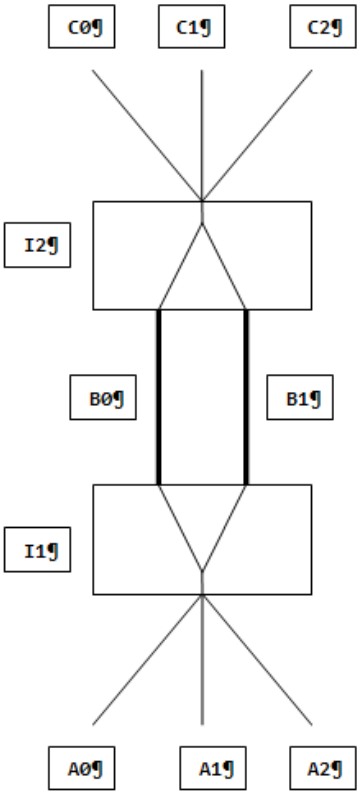
Un carrello è rappresentato dal *thread* **`Carrello`** il cui metodo **`run`** effettua tutta la sequenza di operazioni previste per un carrello (accodamento nella tratta di ingresso impostata su base casuale, prelievo da I1, permanenza nell'elevatore assegnato, prelievo da I2, passaggio alla tratta di uscita corrispondente e allontanamento).

Oltre alla regione di sincronizzazione, si utilizzino (tra gli altri) anche le variabili e vettori seguenti:

- **`I1`**, **`I2`**, che rappresentano lo stato degli omologhi meccanismi;
- **`B[2], C[3]`** che rappresentano lo stato rispettivamente dei due elevatori e delle 3 tratte d'uscita;
- **`ticket[3], serv[3]`** che servono ad implementare il sistema di *ticketing* per mantenere l'ordinamento FIFO nelle 3 code d'attesa in ingresso.

Si simuli la permanenza in I1 e in I2 con un'attesa costante rispettivamente pari a I1ATT e I2ATT, e la permanenza in Bh e in Ci con attese casuali tra due valori, minimo e massimo, rispettivamente BMIN/BMAX e CMIN/CMAX.

N.B. Si suggerisce per esercizio di svolgere il tema anche con uno degli altri costrutti di sincronizzazione (Semafori [privati], Monitor di Hoare e di Java).



COGNOME _____ NOME_____ **LT ◊ LM ◊** MATR. _____

SISTEMI OPERATIVI – Prova scritta

B – Una azienda ad alta automazione effettua due tipi di lavorazione, WORKAB che ha bisogno di un pezzo di tipo A e di un pezzo di tipo B, e WORKA che abbisogna solo di un pezzo di tipo A. I pezzi sono presenti in due magazzini distinti di capienza rispettivamente TOTA e TOTB e i pezzi vengono portati ai magazzini uno alla volta. Diverse stazioni operatrici equivalenti ripetono ciascuna un ciclo costituito, di norma, da 3 lavorazioni successive di tipo WORKAB, seguite da una lavorazione di tipo WORKA. Se per mancanza di pezzi una lavorazione WORKAB non può essere eseguita, la stazione operatrice attende un massimo di MAXT unità di tempo, e quindi, in sostituzione di WORKAB, tenta di eseguire immediatamente una lavorazione WORKA, altrimenti attende di nuovo per un massimo di MAXT per una WORKAB ed eventualmente di nuovo una sostituzione immediata con WORKA. Questa sotto-sequenza si ripete finché una lavorazione non è possibile. Una richiesta per una WORKAB è sempre prioritaria rispetto ad una WORKA ,se entrambi possibili.
Si realizzi in ADA-Java il sistema rappresentato da un thread server che controlla i magazzini e un thread client che fa da modello di simulazione delle stazioni operatrici. Il server **`Magaz`** dovrà implementare i seguenti *entry*:

- **`depA(numA: out int)`**
- **`depB(numB: out int)`**
  richieste di inserimento di un semilavorato, di tipo A o B. Il client di alimentazione deve attendere se il relativo deposito è pieno. Questi *entry* restituiscono il numero di pezzi presenti nel relativo deposito dopo l'aggiunta.
- **`workAB()`**
  richiesta di consenso per una lavorazione WORKAB. Il client attende se non sono disponibili i pezzi necessari.
- **`workA()`**
  richiesta di consenso per una lavorazione WORKA. Il client attende se è possibile una lavorazione WORKAB e c'è una richiesta di quel tipo pendente, oppure se non c'è la disponibilità di un pezzo di tipo A.

Il client **`Oper`**, che rappresenta una stazione operatrice, esegue indefinitamente il ciclo descritto. Simulare la durata delle singole lavorazioni con un'attesa casuale compresa tra WORKMIN e WORKMAX. Non è richiesta la realizzazione del/i thread di produzione dei semilavorati.

N.B. Quest'anno nel secondo tema verrà richiesto solo il codice centrale del server, cioè il selective-wait che include i diversi entry accept con le relative guardie ed eventualmente la chiamata del client