SISTEMI OPERATIVI IDT/IEL: Seconda prova intermedia, 17 giugno 2004

Esercizio 1

Sia dato un sistema che prevede risorse di tipo R_A, R_B, R_C, e R_D, delle quali sono presenti rispettivamente 4, 3, 2, e 1 istanze. Sul sistema si trovano in esecuzione 3 processi (P_1, P_2, e P_3), le cui richieste massime per le suddette risorse sono indicate nella tabella sottostante:

	R_A	R_B	R_C	R_D
P_1	3	3	1	1
P_2	3	2	2	0
P_3	1	1	0	1

All'istante t_0 alcune istanze delle suddette risorse sono allocate ai processi, come indicato di seguito:

	R_A	R_B	R_C	R_D
P_1	2	2	0	0
P_2	1	0	1	0
P_3	0	1	0	1

- Applicando l'algoritmo del banchiere si determini se il sistema si trova in uno stato sicuro, ed eventualmente si individui una sequenza sicura;
- nell'ipotesi che sul sistema sia presente una sola istanza della risorsa R_C (anziché 2) si verifichi se il sistema si trova in uno stato sicuro;
- analogamente, si verifichi se il sistema si trova in uno stato sicuro nell'ipotesi che sul sistema siano presenti 3 istanze della risorsa R_C.

Esercizio 2

Si definisca una classe Java, denominata *Blocco*, che implementi i seguenti metodi

- `attendi()`: se nessun thread ha ancora invocato il metodo `sblocca()` (vedi sotto) sulla particolare istanza della classe, il thread invocante viene sospeso su di essa; altrimenti il metodo ritorna immediatamente e il thread può procedere nella sua esecuzione.
- `sblocca()`: se esistono thread sospesi sull'istanza della classe, questi vengono sbloccati; quindi, il metodo ritorna.

Si definiscano inoltre altre due classi (denominate *Riempimento* e *SommeParziali*) che implementano dei threads. Ciascuna delle due classi richiede che, in fase di istanziazione, siano forniti il riferimento ad un vettore di interi nonché i riferimenti a due oggetti della classe *Blocco*. Per entrambe le classi l'elaborazione si compone di due fasi: l'avvio della prima sarà consentito solo dopo che il primo blocco sarà stato sbloccato; al termine della prima fase, e prima di inizare le seconda, dovrà essere sbloccato il secondo blocco. Il dettaglio dell'elaborazione da svolgersi nelle due fasi per le due classi è descritto nel seguito:

- *Riempimento*: nella prima fase di esecuzione del thread, se N è la dimensione del vettore di interi, questo sarà riempito con valori crescenti da 0 a N-1 ($v[i]=i$). Nella seconda fase si provvederà a stampare sullo schermo i valori precedentemente determinati;
- *SommeParziali*: durante la prima fase dell'elaborazione si sostituirà il valore di un elemento con la somma dei valori originariamente contenuti negli elementi che lo precedono, incluso l'elemento corrente ($v'[i]=\sum_{k=0}^{i} v[i]$). Nella seconda fase il thread provvederà a stampare sullo schermo il contenuto del vettore.

Infine, si realizzi un programma che, dato un vettore di interi di dimensione N, provveda a creare un'istanza ciascuna delle due classi Riempimento e SommeParziali, e quindi ad avviare i rispettivi thread. Una volta avviati entrambi i threads, il programma dovrà avviare il riempimento sbloccando il primo thread. Il secondo thread sarà sbloccato al completamento della prima fase del primo thread. Il programma principale si bloccherà finché il secondo thread non avrà completato la prima fase, dopodiché provvederà a stampare il valore dell'ultimo elemento del vettore (che corrisponde alla somma dei valori contenuti nel vettore dopo il riempimento). Si ponga particolare attenzione al numero di oggetti della classe *Blocco* che devono essere creati per rispettare le specifiche.