DIMES UNIVERSITÀ DELLA CALABRIA
Dipartimento di **Ingegneria Informatica, Modellistica, Elettronica e Sistemistica**

Esame di **Sistemi Operativi** del Corso di Laurea in Ingegneria Informatica DM270

Prova scritta del 18 settembre 2017 – Durata: 2 ore e 30 minuti – Orario di consegna: _____

Cognome e Nome:	Matricola:

Prova 1 (4 punti)

Si consideri la seguente successione di riferimenti a pagine in memoria centrale

1, 2, 3, 4, 1, 3, 5, 6, 2, 4, 2, 5, 3, 1, 6, 3

Indicare quante assenze di pagine (page fault) si verificano rispettivamente se si usano 3 e 4 blocchi di memoria con l'algoritmo di sostituzione ottimale. E' richiesto di riportare per intero il procedimento seguito per ottenere i risultati.

Prova 2 (4 punti)

Si descriva brevemente, anche mediante figure commentate, quale è lo scopo della memoria virtuale e come viene implementata nei sistemi operativi.

Prova 3

Si consideri una funivia che permette di spostare i turisti da un piccolo paese fino alla cima della montagna. La funivia può essere occupata da 6 **turisti che sono arrivati lì a piedi** o da 3 **turisti in bici** (i tre posti rimanenti sono occupati dalle bici).

La funivia è guidata da un **pilota** che continuamente sale e scende dalla montagna. Il pilota una volta arrivato a valle fa entrare nella funivia un gruppo di turisti a piedi oppure un gruppo di turisti in bici. Non potranno mai salire sia turisti a piedi sia turisti in bici. Il pilota usa una politica *round-robin*: fa salire un gruppo di 6 turisti a piedi, poi un gruppo di 3 turisti in bici e così via.

La funivia parte solo dopo aver raggiunto il pieno carico (o 6 turisti a piedi o 3 turisti in bici) impiegando 5 minuti per giungere in cima. Il pilota, una volta arrivato in cima, lascia i turisti e ritorna con la funivia vuota a valle impiegando 2 minuti (i turisti scenderanno a piedi o in bici dalla montagna a valle).

Si modelli il sistema descritto in Java, dove il *pilota* e i *turisti* sono dei thread che interagiscono tramite un oggetto *Funivia* che espone (almeno) i seguenti metodi:

- **void pilotaStart:** il pilota ha portato la funivia a valle e attende che salgano i turisti secondo l'ordine specificato sopra. Quando la funivia è piena, il pilota blocca gli accessi e inizia il viaggio.
- **void pilotaEnd**: Il pilota è arrivato in cima, stampa l'ID dei turisti presenti nella funivia e il loro tipo e dopo permette ai turisti di scendere dalla funivia. Subito dopo inizia il ritorno a valle.
- **void turistaSali (int t):** il turista di tipo **t** (***0*** turista a piedi, ***1*** turista in bici) è pronto per salire in montagna. Il turista viene sospeso fin quando non occupa un posto all'interno della funivia. I turisti in fila vengono risvegliati secondo un ordine causale.
- **void turistaScendi (int t)**: permette al turista di tipo **t** di scendere dalla funivia.

Prova 3a (13 punti)

Si implementi la classe Funivia (astratta), Turista e Pilota, e una soluzione che riproduca il funzionamento del problema sopra descritto utilizzando la classe **Semaphore** (usare solo i metodi *acquire* e *release*) del package **java.util.concurrent**. Si scriva anche un ***main*** d'esempio che faccia uso di questa soluzione. Il ***main***, dopo aver definito la Funivia e avviato il pilota, esegue 18 turisti a piedi e 9 turisti in bici.

Prova 3b (9 punti)

Si implementi una soluzione che riproduca il funzionamento del problema sopra descritto utilizzando gli strumenti di mutua esclusione e sincronizzazione del package **java.util.concurrent.locks**.

N.B.: 1): Si chiede di commentare il codice e/o aggiungere delle stampe a video. 2): Si chiede di scrivere un breve commento per ogni semaforo/condition usato. 3) Si prega di numerare i fogli dove sono svolti gli esercizi. 4) Si prega di usare una calligrafia leggibile.