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 26 gennaio 2016 – Durata: 2 ore e 30 minuti – Orario di consegna: _____

Cognome e nome:	Matricola:

Prova 1 (4 punti)

Si supponga di disporre di un disco composto da 2000 cilindri numerati da 0 a 1999. Il disco sta servendo una richiesta relativa al cilindro 200 e la richiesta precedente era relativa al cilindro 280, la coda di richieste inevase in ordine FIFO è composta delle seguenti richieste:

1400, 150, 800, 500, 1200, 1500, 400, 900, 180

assumendo come punto di partenza la posizione attuale della testina, calcolare la distanza totale (in cilindri) che il braccio del disco percorre per soddisfare tutte le richieste inevase usando l'algoritmo di scheduling SSTF.

Prova 2 (4 punti)

Si descriva, anche mediante figure opportunamente commentate, il funzionamento dello schema di allocazione concatenata dei blocchi dei file di un File System sul disco.

Prova 3 (22 punti)

Si vuole realizzare un sistema per gestire una gara di **snowboard**. Alla gara partecipano **N snowboarder**, ognuno dei quali è dotato di un numero di maglia che va da **0** a **N-1**. Gli snowboarder vengono fatti partire uno per volta da **un addetto di gara** ed impiegano da 1 a 5 minuti per arrivare al traguardo. L'addetto di gara dopo che lo snowboarder ha tagliato il traguardo stampa su schermo il tempo di discesa e la posizione temporanea in classifica (ad esempio 2,46 min. e "3°"). Dopo che l'ultimo snowboarder ha tagliato il traguardo, l'addetto di gara stamperà su schermo la classifica finale, ovvero i numeri di maglia degli snowboarder ordinati da quello che ha impiegato meno tempo a quello che ne ha impiegato di più (ad esempio "15 18 3 22...").

Si modelli il sistema descritto in Java, dove gli *snowboarder* e l'*addetto di gara* sono dei thread che interagiscono tramite un oggetto *Gara* che espone solo i seguenti metodi:

- **`void partenza(Snowboarder s)`**: sospende lo snowboarder **`s`** fin quando non è il suo turno (ordine FIFO di arrivo) e la pista non è libera.
- **`int arrivo(Snowboarder s)`**: permette allo snowboarder **`s`** di tagliare il traguardo;
- **`boolean stampaEprossimo()`**: l'addetto stampa le informazioni riguardo l'ultimo snowboarder che ha usato la pista (tempo di discesa e posizione temporanea dello snowboarder) e fa partire il prossimo snowboarder. Quando gli snowboarder sono terminati restituisce false;
- **`void classificaFinale()`**: L'addetto stampa su schermo la classifica finale.

Si implementino due soluzioni che riproducano il funzionamento del problema sopra descritto utilizzando:

- gli strumenti di mutua esclusione e sincronizzazione del package java.util.concurrent.locks;
- la classe Semaphore (usare solo i metodi acquire e release) del package java.util.concurrent

Si scriva infine un main d'esempio che, facendo uso di una delle due soluzioni precedenti, inizializzi un oggetto Gara, inizializzi `N=50` snowboarder e l'addetto, e ne avvii l'esecuzione.