*Esame di **Sistemi Operativi** del Corso di Laurea in Ingegneria Informatica DM270*
*Prova scritta del **16 luglio 2013** – Durata: 2 ore e 45 minuti – Orario di consegna:*

Cognome e nome:	Matricola:

Prova 1

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 700 e la richiesta precedente era relativa al cilindro 90, la coda di richieste inevase in ordine FIFO è composta delle seguenti richieste:

400, 100, 200, 50, 1500, 300, 1400, 1200, 650

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 i seguenti algoritmi di scheduling:

1. SSTF;
2. C-SCAN.

Prova 2

Si descriva, anche con l'aiuto di schemi e/o grafici opportunamente commentati, come si traduce un indirizzo logico in indirizzo fisico nella paginazione.

Prova 3

Una partita di pallacanestro viene disputata da due squadre composte da cinque giocatori ciascuna. All'inizio della partita un giocatore a caso delle due squadre possiede la palla mentre gli altri sono in attesa di riceverla. Il giocatore che possiede la palla la tiene per un tempo che varia da **1 a 5 secondi**, dopodiché effettua un passaggio o un tiro a canestro in funzione del numero di passaggi consecutivi effettuati dalla propria squadra: se ne ha effettuati tre allora prova il tiro a canestro, altrimenti effettua un passaggio.
Ogni volta che un giocatore di una squadra effettua con successo un passaggio, o un tiro a canestro, la palla passa ad uno qualsiasi dei giocatori della propria squadra. Ogni tiro a canestro effettuato con successo porta alla squadra che lo ha realizzato un punto. Quando un giocatore di una squadra fallisce un passaggio, o un tiro a canestro, la palla passa ad uno qualsiasi dei giocatori della squadra avversaria. Ogni giocatore ha una propria probabilità di riuscita del passaggio/tiro a canestro compresa tra il **30% e il 60%**, fissata in fase di inizializzazione.
Alla partita partecipa anche un arbitro il cui solo compito è quello di terminare il gioco allo scadere dei **40 minuti** dall'inizio della partita e stamparne in uscita il risultato. La partita termina comunque anche in caso di parità. Quando la partita termina tutti i giocatori e l'arbitro terminano la propria esecuzione.

Si modelli il sistema descritto in Java, dove i **giocatori** e l'**arbitro** sono dei thread che interagiscono tramite un oggetto **partita**: tale oggetto espone tre metodi, (1) `riceviPalla(int s)`, (2) `lanciaPalla(int s, int p)` che permettono al giocatore della squadra *s* rispettivamente di ricevere la palla e di passare o tirare a canestro la palla con probabilità di successo pari a *p*, e (3) `termina()`, che permette all'arbitro di terminare la partita. I metodi (1) e (2) restituiscono un booleano: `true` se la partita è in corso, `false` se la partita è terminata. Si implementino due soluzioni che riproducano il funzionamento del problema sopra descritto utilizzando:

- la classe `Semaphore` del package `java.util.concurrent`
- gli strumenti di mutua esclusione e sincronizzazione del package `java.util.concurrent.locks`

Si scriva infine un main d'esempio che, facendo uso di una delle due soluzioni precedenti, inizializzi una **partita**, inizializzi 10 **giocatori**, 1 **arbitro** e ne avvii l'esecuzione.