Esame di Informatica II – Modulo Sistemi Operativi, 14/06/2016

Nome: _____ Cognome: _____ Matricola: _____

Domanda 1:

Struttura monolitica, stratificata e a micro kernel. Descrivere brevemente i principi, i pregi e i difetti di ciascuna di queste tecniche di progettazione dei sistemi operativi.
Facoltativo: indicare anche ragioni storiche ed esempi di implementazione delle tecniche elencate.

Nome: _____ Cognome: _____ Matricola: _____

Domanda 2:

Considerare un sistema in cui vengono attivati i processi descritti nella seguente tabella:

Nome	istante di attivazione (unità di tempo)	Durata (unità di tempo)
P1	0	7
P2	2	4
P3	4	4
P4	5	1

Assumere che il sistema sia equipaggiato con una singola CPU. Condurre tre simulazioni del comportamento di uno scheduler a breve termine, immaginando che esso operi rispettivamente con politica:
1) First Come First Served,
2) Shortest Job First (non preemptive),
3) Round Robin (quanto = 5 unità di tempo).
Per ogni simulazione riportare:
- il diagramma Gantt che descrive come la CPU è allocata ai processi,
- il calcolo del tempo medio di attesa dei processi,
- il calcolo del tempo medio di completamento (turnaround) dei processi.

Nome: _____ Cognome: _____ Matricola: _____

Domanda 3:

Data la seguente porzione di codice, che rappresenta una parte dell'implementazione della classe BoundedBuffer, completare il codice del costruttore e fornire un'implementazione dei metodi:
- insert: consente di inserire un nuovo elemento all'interno del buffer, se non è pieno;
- remove: consente di rimuovere un elemento dal buffer, se non è vuoto.

L'implementazione fornita dovrà garantire che le operazioni effettuate sulla struttura dati values avvengano in mutua esclusione. Inoltre, un ipotetico thread Consumer deve rimanere in attesa se il buffer è vuoto fino a che non viene effettuata una insert. Un ipotetico thread Producer, invece, deve rimanere in attesa se il buffer è pieno fino a che non viene effettuata una remove. Come meccanismo di sincronizzazione, utilizzare la classe Semaphore del package java.util.concurrent.

```
import java.util.ArrayList;
import java.util.List;
import java.util.concurrent.Semaphore;

public class BoundedBuffer<T> {

	private List<T> values;
	private int bufferSize;
	// posizione corrente di inserimento-rimozione elementi
	private int count;

	// semaforo binario per la mutua esclusione
	private Semaphore mutex;
	// semaforo per regolare l'accesso dei produttori
	private Semaphore empty;
	// semaforo per regolare l'accesso dei consumatori
	private Semaphore full;

	public BoundedBuffer(int items) {
		bufferSize = items;
		count = 0;
		values = new ArrayList<T>(bufferSize);
		// TODO: inizializzazione dei semafori

	}
```

```
        @Override
        public void insert(T item) {
            // TODO: implementazione dell'operazione
            // inserimento thread safe









        }

        @Override
        public T remove() {
            // TODO: implementazione dell'operazione
            // rimozione thread safe









        }
}
```