**Sistemi Operativi**

Compito 25-01-2021

**Esercizio 1 (10 punti)**

Un sistema operativo adotta la politica di scheduling dei thread a code multiple con tre code e prelazione tra le code, la coda Q1 ad alta priorità con scheduling round robin (q=1ms), la coda Q2 a media priorità con scheduling round robin (q=2ms), la coda Q3 a bassa priorità con scheduling FCFS. Se un thread viene prelazionato viene messo in fondo alla coda.

Il sistema deve schedulare i seguenti thread arrivati ai tempi indicati nelle code indicate e con uso CPU/IO indicati:

| | | | |
|---|---|---|---|
| $T_1$ | coda Q1 | $T_{arr}$=2 | CPU(2ms)/IO(3ms)/CPU(2ms) |
| $T_2$ | coda Q2 | $T_{arr}$=3 | CPU(1ms)/IO(3ms)/CPU(2ms) |
| $T_3$ | coda Q2 | $T_{arr}$=1 | CPU(2ms)/IO(3ms)/CPU(3ms) |
| $T_4$ | coda Q3 | $T_{arr}$=0 | CPU(5ms) |

Si determini: il **diagramma di Gantt**, il **tempo di attesa** medio, il **tempo di ritorno** medio, il **tempo di risposta** medio, il numero di cambi di contesto.

**Esercizio 2 (20 punti)**

Sono presenti N thread *Generator* che iterativamente acquisiscono un valore da un contatore condiviso (due thread non possono ottenere lo stesso valore) e inseriscono in una coda un messaggio formato dal valore del contatore (campo *t*) e un valore casuale (campo v) e quindi aspetta un tempo *TG* prima di generare un nuovo messaggio. Sono presenti M thread *Consumer* che iterativamente prelevano dalla coda condivisa il messaggio con il valore del campo *t* minore, stampano il messaggio e aspettano un tempo *TC* prima di prelevare un nuovo messaggio. Il programma principale deve creare i thread, quindi deve attendere 10 secondi, terminare i thread Generator e quindi attendere che la coda sia vuota e quindi terminare i thread *Consumer* e infine stampare il valore del contatore (il suo valore deve corrispondere all'ultimo messaggio stampato).

Realizzare in Java il sistema descritto usando i semafori per la sincronizzazione tra thread.