UNIVERSITÀ DEGLI STUDI DI BOLOGNA – CORSO DI LAUREA IN INFORMATICA
CORSI DI SISTEMI OPERATIVI A-L e M-Z. PROGRAMMAZIONE CONCORRENTE
ANTICIPO SESSIONE ESTIVA – ANNO ACCADEMICO 2001/2002
17 GENNAIO 2002

Esercizio -1: essersi iscritti correttamente per svolgere questa prova.
Esercizio 0: Scrivere correttamente il proprio nome, cognome e numero di matricola in ogni foglio prima di svolgere ogni altro esercizio seguente.
Esercizio 1: Il comportamento dei gestori di interrupt puo' essere esaminato come un problema di programmazione concorrente.
Un dispositivo quando deve segnalare un evento alla CPU (e.g. fine I/O) manda un interrupt mediante una chiamata:

```
intmodel.interrupt(level)
```

dove `level` è il livello di priorità dell'interrupt.
Il kernel puo' cambiare il livello di gestione degli interrupt mediante la chiamata

```
intmodel.newlevel(level)
```

`newlevel(n)` significa che tutti gli interrupt di livello minore di `n` devono essere mascherati e il loro effetto differito.
Un gestore di interrupt (*interrupt handler*) viene visto come un processo così strutturato

```
handler[k]: process
while (true) {
intmodel.wait4int(k);
// .... gestisci interrupt
}
}
```

C'è un gestore per ogni livello di priorità, la gestione dell'interrupt deve essere fatta in mutua esclusione (uno alla volta), se ci sono più interrupt pendenti viene gestito quello di priorita' massima giunto per prim , la gestione non prevede preemption. Scrivere il monitor `intmodel`.

Esercizio 2: Scrivere il modulo (classe) intmodel facendo uso di Regioni Critiche Condizionali (CCR)

Esercizio 3: Siano date le funzioni DP e DV definite come segue:

```
s.DP() {
  s.P();
  s.P();
}
```

```
s.DV() {
  s.V();
  s.V();
}
```

Sono equivalenti alle normali operazioni sui semafori? perché?