Sistemi Operativi (modulo II / B) **CON SOLUZIONE** 23 maggio 2018

Esercizio 1

Si consideri una variante del problema dei filosofi in cui i filosofi raccolgono due bacchette qualsiasi (il semaforo `bacchetta` è inizializzato a 5):

```
thread filosofo(i) {
  while(true) {
    // pensa
    P(bacchetta); // raccoglie una bacchetta
    P(bacchetta); // raccoglie una bacchetta
    // mangia
    V(bacchetta); // deposita una bacchetta
    V(bacchetta); // deposita una bacchetta
  }
}
```

1. Discutere la possibilità di stallo tra i filosofi mostrando una possibile esecuzione problematica
   Come nel problema standard, se tutti i filosofi raccolgono una bacchetta, non ci saranno più bacchette disponibili (il semaforo `bacchetta` sarà rosso) e si creerà una situazione di attesa circolare irrisolvibile (stallo). Notare che è l'unica situazione che crea un'attesa circolare in quanto, se uno dei filosofi raccoglie 2 bacchette, prima o poi le restituirà.

2. Proporre una modifica del codice che prevenga lo stallo, spiegandone il funzionamento e cercando di non sincronizzare in modo eccessivo i filosofi
   E' sufficiente aggiungere un secondo semaforo `sedie`, inizializzato a 4, che permetta di mangiare solamente a 4 filosofi. Basterà invocare `P(sedie)` e `V(sedie)` rispettivamente prima e dopo le due `P(bacchette)`. In questo modo, non è più possibile che 5 filosofi invochino la prima `P(bacchetta)` perché l'ultimo verrebbe bloccato su `P(sedie)`. L'attesa circolare non si forma più.

Esercizio 2

Progettare un Monitor `S` che sincronizzi i thread seguenti in modo da stampare MONITOR:

```
thread PRO {
   print "MO";
   S.notifica("MO");
}
```

```
thread MO {
   S.attendi("MO");
   print "NI";
   S.notifica("NI");
}
```

```
thread SSO {
   S.attendi("MONI");
   print "TOR";
}
```

Scrivere lo pseudocodice del Monitor illustrandone il funzionamento (**fare attenzione ai parametri** passati ai metodi `notifica` e `attendi`):

```
Monitor S {
  String testo = ""; // testo complessivo stampato, inizialmente vuoto

  void notifica(String s) {
    testo = testo + s;  // aggiunge s al testo stampato
    notifyAll();        // notifica tutti i thread (non usiamo condition)
  }
  void attendi(String s) {
    while(testo != s) // attende che sia stato stampato s
      wait();
  }
```

Il monitor proposto utilizza la stringa `testo` per memorizzare il testo stampato dai thread. Non utilizziamo variabili condition (come in Java) e quindi notifichiamo tutti i thread con `notifyAll`. Il metodo `notifica` aggiunge a `testo` la stringa notificata `s` e sveglia eventuali thread in attesa con una `notifyAll()`. Il metodo `attendi`, mette il thread in attesa finché l'intero testo `s` non sia stato stampato dai thread.