Esercizio 1

Utilizzare i semafori per sincronizzare i seguenti thread in modo che stampino DOVEVO. (Inserire le P e le V direttamente nel codice qui sotto):

```
thread D {
   P(SemD)
   print "D";
   V(SemO);

}
```

```
thread O {
  for(i=0;i<2;i++) {
   P(SemO);
   print "O";
   if (i==0) V(SemV);

  }
}
```

```
thread V {
  for(i=0;i<2;i++) {
   P(SemV);
   print "V";
   if (i==0) V(SemE);
   else V(SemO);
  }
}
```

```
thread E {

   P(SemE);
   print "E";
   V(SemV);

}
```

Semafori e valori di inizializzazione: I semafori SemD, SemO, SenV, SemE sono inizializzati rispettivamente a 1,0,0,0 e regolano la stampa della rispettiva lettera.

Breve spiegazione della soluzione proposta: L'unico thread che può stampare è D in quanto il semaforo corrispondente (SemD) è verde. Dopo ogni stampa viene abilitato il thread successivo. Nel caso della D e della E il thread successivo è, rispettivamente, O e V. Nel caso del thread O non si deve sbloccare nulla quando i == 1, perché è l'ultima lettera a essere stampata. Il thread V sblocca E (la E di DOVE) alla prima iterazione e O (la O di DOVEVO) alla seconda.

Esercizio 2

Considerare il monitor `tavola` che alloca le risorse `bacchette[i]` ai thread `filosofo(i)` che ne fanno richiesta.

```
Monitor tavola {
  bacchette[5] = {true,true,true,true,true};

  void richiedi(int i) {
    while (!bacchette[i]) wait();
    bacchette[i] = false;
  }
}
```

```
void rilascia (int i) {
  bacchette[i] = true;
  notifyAll();
}
```

Discutere la possibilità di stallo nel caso cinque thread `filosofo(i)` (con `i` = 0 … 4) invochino `tavola.richiedi(i); tavola.richiedi((i+1)%5)` simultaneamente, e proporre una soluzione a tale problema in cui i filosofi raccolgono le due bacchette in modo atomico, scrivendo il relativo codice.

E' il problema classico dei filosofi a cena: se tutti i filosofi si allocano la prima bacchetta i-esima rimarranno tutti in attesa della bacchetta (i+1)%5. E' una situazione di attesa circolare irrisolvibile, ovvero uno stallo.

Una possibile soluzione è far raccogliere entrambe le bacchette tramite una singola invocazione al metodo `richiediAtomico(i)`, che attende finché entrambe le bacchette non sono disponibile e, quando disponibili, le alloca entrambe. Tale metodo può essere implementato nel modo seguente (il rilascio può comunque avvenire tramite due invocazioni di rilascia, come nel caso precedente)

```
Monitor tavola {
  bacchette[5] = {true,true,true,true,true};

  void richiediAtomico(int i) {
    while (!bacchette[i] || !bacchette[(i+1)%5])
wait();
    bacchette[i] = false;
    bacchette[(i+i)%5] = false;
  }
}
```

```
void rilascia (int i) {
  bacchette[i] = true;
  notifyAll();
}
```