Esercizio 1

Si consideri una variante del problema dei filosofi in cui i filosofi raccolgono due bacchette scelte a caso:

```
thread filosofo(i) {
  while(true) {
    // pensa
    sceglie i e j a caso tra 0 e 4, con i != j
    P(bacchetta[i]); // raccoglie la bacchetta i-esima
    P(bacchetta[j]); // raccoglie la bacchetta j-esima
    // mangia
    V(bacchetta[i]); // deposita la bacchetta i-esima
    V(bacchetta[j]); // deposita la bacchetta j-esima
  }
}
```

1. Discutere la possibilità di stallo tra due filosofi mostrando una possibile esecuzione problematica
   Se ad esempio un filosofo sceglie i=1 e j=3 e un secondo filosofo sceglie i=3 e j=1, nel caso raccolgano simultaneamete la prima bacchetta (1 e 3) andranno in stallo sulla seconda bacchetta, in quanto si forma una situazione di attesa circolare: il primo filosofo possiede 1 e attende 3, il secondo filosofo possiede 3 e attende 1.

2. Proporre una modifica del codice che prevenga lo stallo, senza modificare la scelta casuale delle bacchette
   E' sufficiente che le bacchette vengano sempre raccolte in ordine crescente, come stabilito dalla allocazione gerarchica, una tecnica studiata a lezione che previene lo stallo. Nel nostro caso basta scambiare i e j nel caso j sia più piccolo di i:

```
if (j<i) {
  tmp = j;
  j = i;
  i = tmp;
}
```

Esercizio 2

Il thread `main` deve attendere l'esecuzione di tutti i thread `task(i)` con un identificativo i pari. Quando `task(i)` termina invoca `m.terminato(i)`, mentre il thread `main` invoca `m.attendiPari()`. Il numero totale dei task (pari e dispari) è `N_TASK=100` (un numero pari arbitrario). Realizzare il monitor `m` che implementi le opportune sincronizzazioni:

```
Monitor m {
  int N_TASK = 100; // numero totale dei task
  int pterminati=0; // task pari terminati

  void terminato(int i) {
    if (i%2 == 0)
      pterminati++;
    if (pterminati == N_TASK/2)
      notify();
  }
}
```

```
void attendiPari () {
  while (pterminati < N_TASK/2)
    wait();


}
```

Spiegare brevemente la soluzione proposta: E' sufficiente contare quanti thread pari terminano (variabile `pterminati`) e attendere finché `pterminati` non diventa uguale a `N_TASK/2`. Inizializziamo la variabile pterminati a 0 e la incrementiamo, nella funzione terminato, quanto un thread pari termina (quanto i%2 == 0). Se la variabile assume valore N_TASK/2 notifichiamo l'eventuale thread in attesa. Nella funzione attendiPari attendiamo finché pterminati < N_TASK/2.