# atomiche-random2

Sia data la funzione ransum così definita

ransum(x,y,z) = <y=x+random(4,-8,-2), x=z*random(1,-3,5) >

dove random ogni volta restituisce casualmente uno dei parametri passati;

Tramite questa istruzione, è possibile implementare un meccanismo di sezioni critiche simile a quello in cui si usa Test&set? In caso positivo, spiegare come. In caso negativo, spiegare perché.

Soluzione
Sì. Per dimostrare che questo è vero, bisogna mostrare un algoritmo e dimostrare che soddisfa le proprietà richieste.

```
turn = 1000; COBEGIN P(1) // P(2) // ... // P(n) COEND
P(i): process
  while(true) {
    myturn = 0;
    while (myturn < 500) {
      ransum(turn, myturn, 0);
    }
    /* critical section */
    turn = 1000;
    /* non-critical section */
  }
end
```

**Lemma**: turn puo' assumere solo i valori 0 oppure 1000.
Dim: Inizialmente, turn = 1000; tutte le volte che si termina una critical section, turn e' ancora 1000. Quando si esegue ransun, turn prende il valore di z=0 per un numero casuale, quindi prende il valore 0.

**Lemma**: myturn puo' assumere solo valori compresi fra -8 e +4 o valori tra 992 e 1004.
Dim: se turn e' uguale a 0, myturn prendera' un valore tra -8 e +4; se turn e' uguale a 1000, myturn prendera' un valore tra 992 e 1004. Non si danno altri casi.

**Lemma**: l'algoritmo soddisfa mutua esclusione

Supponiamo per assurdo che due processi (chiamiamoli P e Q) riescano ad entrare entrambi nella mutua esclusione.
- Senza perdere in generalita', supponiamo che P entri per

primo.
- Poiche' e' entrato, ha eseguito almeno una istruzione ransum
- quindi dopo l'istruzione ransum, turn e' uguale a 0
- anche Q deve aver eseguito un'istruzione ransum, dopo quella  eseguita da P; quindi il valore di myturn di Q e' < 500;
  assurdo.

Lemma: l'algoritmo soddisfa assenza di deadlock

supponiamo che tutti i processi non riescano ad entrare, perche' in deadlock; questo significa che turn = 0, altrimenti il primo ad eseguire ransum sarebbe passato. poiche' turn e' 0, qualche processo deve essere dentro la sezione critica, perche' una volta fuori, mette turn a 1000. Assurdo

Lemma: l'algoritmo soddisfa assenza di starvation?

no; come altre soluzioni test&set

Lemma: l'algoritmo soddisfa assenza di ritardi inutili

si; se P e' l'unico processo a voler entrare nella sezione critica, P trovera' turn = 1000 e potra' entrare.