**Tema 1**

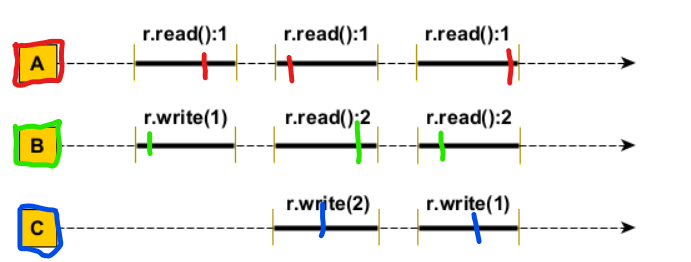
# **Componența echipei:**

**Inculeț Alexandru(TPM1)**

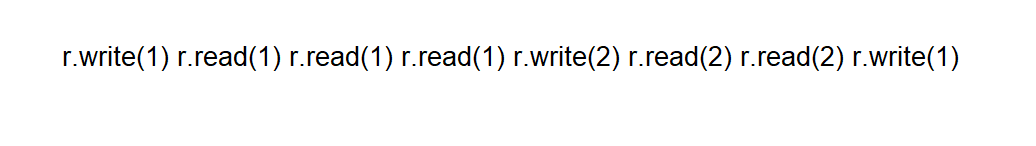
**Inculeț Bogdan(TPM1)**

**Ioan Alexandra(TPM1)**

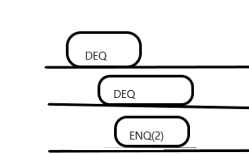
1. Secvența este liniarizabila, fiind posibil să stabilim punctele de linearizare după cum urmează:



De asemenea, secvența este consistent secvențială, deoarece putem reordona threadurile paralele astfel încât secvența să respecte regulile FIFO după cum urmează:



1. **a.**  Generalizarea propusă nu funcționează corect. Dacă am avea coada vidă și am aplica simultan, de exemplu, două threaduri cu deq() și un thread cu enq(x), odată ce se va băga elementul x în coadă, cele două threaduri vor fi active simultan și vor scoate elementul, apoi vor încerca să scoată din coada vidă, rezultând într-o eroare.



**b.** Al doilea element din tuplele care sunt comparate reprezinta indexul threadului, aceste este unic pentru fiecare thread, este necesar ca acesta sa faca parte din comparatie pentru a asigura o ultima forma de prioritate **sigură** a threadului cu index mai mic fata de alte threaduri cu acelasi label, dar index mai mare si pentru a limita accesul multiplor threaduri la zona critica in acelasi timp.

De exemplu fie threadurile A, B, si C cu indecsii 0, 1, 2 si toate cu acelasi label : 5. Daca nu ar fi luat in considerare si indexul threadului atunci A,B,C nu ar astepta ca sa se elibereze lockul si ar avea toate acces simultan la zona critica.

**c.** La utilizarea unui lock se preferă ca apelul lock() să fie executat înainte de blocul try, și nu în cadrul acestuia, deoarece în caz contrar, o excepție care ar putea apărea la lock() ar conduce la apelarea incorecta a unlock().