Tema 2

Realizatori: Horodincă Mihai și Mirilă Vasile Dănuț

1. a. Linia de cod care reprezinta punct de linearizare pentru adaugare cu succes in lista este “pred.next = node” deoarece se modifica structura listei.

b. Linia de cod care reprezinta punct de linearizare pentru esec la adaugare element in lista este “return false” deoarece aceasta linie este executata atunci cand elementul este deja in lista.

c. Linia de cod care reprezinta punct de linearizare pentru stergere element din lista este “pred.next = current.next” deoarece se modifca structura listei.

d. Linia de cod care reprezinta punct de linearizare pentru esec la stergere element din lista este “return false” deoarece aceasta linie este executata atunci cand elementul nu se gaseste in lista.

2. a. In metoda “enq” nu este necesara plasarea “size.getAndIncrement()” in cadrul sectiunii protejate de “enqLock” deoarece “size” este o variabila atomica, deci la apelarea metodei “getAndIncrement” se va obtine un lock pe variabila “size”, astfel aceasta poate fi modificata doar de catre un singur thread. Astfel nu vor fi comportamente nedorite.

b. In acest caz, in metoda “enq” va fi lock doar pe nodul head. Astfel pot veni mai multe thread-uri care vor sa adauge cate un element in coada si este posibil ca un nod (tail) sa aiba mai multi descendenti. Deci caracterul FIFO nu ar mai fi pastrat.

2 c. Coadă cu spinning

Aceasă abordare, per ansambu, rezultă într-o funcționare corectă a cozii. În continuare, un singur thread va avea acces la secțiunea protejată de lacăte ceea ce înseamnă că nu vor apărea situații în care două thread-uri să introducă/elimine un element în același timp. O diferență față de varianta cu await este faptul că threadul care a luat inițial lock-ul nu va renunța la el și este sigur că el își va continua rularea după ce există elemente de luat/pus.

Coadă amestec

Și această variantă funcționează. Chiar dacă există mai multe thread-uri care așteaptă în await să existe elemente de luat, spre exemplu, ele vor renunța la lock în această perioadă iar primul thread care pune un element va putea lua acel lock pentru a le notifica.

2 d. Verificarea pentru coadă nenulă trebuie să fie în secțiunea protejată de lacăt pentru că, dacă nu ar fi, am putea da de următoarea situație: Să presupunem că în coadă există doar un element și avem două thread-uri care vor în același timp să scoată câte un element. Ambele vor trece de verificare, unul va lua lock-ul, va lua singurul element iar când lasă lock-ul și îl ia următorul, al doilea va ajunge să ia un pointer NULL, comportament la care nu vrem să ajungem.