Examen - Implementarea Concurenței în Limbajele de Programare

Opţional - Anul III Informatică, Anul IV CTI, 28 ianuarie 2025

Materiale permise: materialele de curs și cele de laborator, inclusiv fișiere cod cu soluții ale problemelor rezolvate în curs și laborator, în anul universitar curent! Alte fișiere nu sunt acceptate.

În timpul examenuli NU AVEȚI VOIE SĂ AVEȚI BROWSERE DESCHISE (cu excepția celor în care rulați cod - onlinegdb.com și onecompiler.com), NU PUTEȚI CONSULTA MATERIALE ONLINE. Nu puteți folosi niciun generator de cod (nici LLM-uri, nici Copilot).

Fiecare student va avea un folder pe care îl va denumi "nume-student-ICLP-2025" în care va avea aceste materiale și în care va rezolva exercițiile. În acest folder, pe parcursul examenului NU pot să apară alte fișiere în afara celor menționate.

La finalul examenului veți face submisia subiectelor rezolvate NUMAI în prezența unui supraveghetor. Vă vom anunța în timpul examenului modul de submitere.

Veti trimite fisierele / arhivele cu solutii pe adresele bogdan.macovei@unibuc.ro SI bogdan.macovei.fmi@gmail.com. Subiectul mail-ului va fi ICLP, Nume Prenume, Grupa. Timpul de transmitere este suplimentar timpului efectiv de 2h alocat pentru rezolvarea subiectelor.

1 Subiectul I - Java/Python/Go (maxim 3p)

a. Sa se implementeze un scenariu *Producer-Consumer* care are un *buffer* de o dimensiune fixa, specificata la creare. Un producator isi va putea executa actiunea atunci cand gaseste loc liber in *buffer*, si se va bloca atunci cand *buffer*-ul este plin.

b. In scenariul de la punctul a., producatorii pot produce obiecte de doua tipuri, iar consumatorii pot consuma doar obiecte de tipul lor - fiecare consumator are, ca informatie interna, tipul obiectelor pe care le poate consuma. Tipurile obiectelor pot fi codificate printr-o informatie numerica - de exemplu tipurile 0 si 1.

In cadrul acestui subiect, nu trebuie ca cele doua roluri sa execute efectiv o metoda, este suficient sa afisati mesaje specifice si sa simulati executarea lor utilizand `Thread.sleep`.

2 Subiectul al II-lea - Haskell (maxim 3p)

Implementati un canal de comunicare care sa functioneze ca o lista de canale de comunicare marginite. Fiecare canal marginit va avea o limita `N` (aceeasi pentru toate canalele din lista), data ca parametru in configurare. Modul de functionare va fi urmatorul:

- pentru citirea din canal, se va specifica indexul din lista al acestuia. Daca indexul nu exista, se va afisa un mesaj corespunzator, altfel se va citi si se va elimina informatia. Citirea dintr-un canal gol nu este blocanta, si va avea ca efect afisarea unui mesaj la `STDOUT`;
- pentru scrierea in canal, se va specifica indexul canalului in care se scrie. Daca indexul nu exista, se va afisa un mesaj de eroare corespunzator. Daca indexul exista, dar s-a depasit capacitatea fixata initial, `N`, atunci apelul va fi blocant. Altfel, daca indexul exista si canalul nu este la capacitate maxima, informatia va fi pusa pe canalul de la indexul specificat.

La crearea canalelor, fiecare va avea cel putin un element.

3 Subiectul al III-lea - Erlang (maxim 3p)

Intr-o casa exista mai multe pisici care sunt hranite prin intermediul unui *feeder*. Stim ca o pisica poate sau sa astepte sa primeasca de mancare (*waiting*), sau sa manance (*eating*), iar *feeder*-ul poate fi ori intr-o stare de asteptare, *idle*, in care primeste cererile din partea proceselor pisica, ori intr-o stare activa, *feeding*, in care hraneste o singura pisica. *Feeder*-ul trebuie sa distribuie mancare pe rand, in ordinea cererilor, iar o pisica poate sa manance doar un interval de 3 secunde. Implementati acest scenariu, utilizand modelul actorilor, si considerati o simulare cu 5 pisici care trebuie hranite.