EXERCITII CURS/LABORATOR

1. a) Implementati modelul **Reader-Writer** definit in cursul 6 (folosind **MyRWLock**).
   b) Implementati o varianta in care numarul de cititori este limitat. Puteti rescrie implementarea sau puteti define un tip de date nou combinand implementarea standard cu un semafor **QSem** (definit in cursul 5).

2. Reamintiti-va implementarea canalelor folosind **MVar**, asa cum este definita in Cursul 6 si definit un nou tip de date pentru canale marginite folosind tipul de date **Chan** si un semafor **QSem**. Verificati implementarea pe un exemplu concludent.

3. a) Rulati programul care citeste dintr-un fisier link-uri la pagini web si le descarca in mod sincron (cursul 6). Puteti inlocui descarcarea paginilor web cu o alta actiune care durata mare, de exemplu calculul numerelor Fibonacci. Rulati folosind optiunea **GHCi :set +s** pentru a vedea timpul de executie. Schimbati programul astfel incat actiunile sa fie executate pe thread-uri separate.
   b) Modificati programul astfel incat sa semnalam momentul in care primul thread si-a terminat executia. Pentru aceasta puteti folosi o variabila **MVar** comuna tuturor thread-urilor. Dupa ce anuntati terminarea primei executii , trebuie sa va asigurati ca si celelalte thread-uri isi termina executia.
   c) Folosind tipul de date **Async** definit in cursul 6, schimbati programul astfel incat actiunile sa fie executate asincron. Scrieti o functie **waitAllMV** care asteapta terminarea unei liste de actiuni asincrone si folositi-o pentru a va asigura ca thread-urile isi termina executia.

4. Rulati exemplul cu conturile bancare in implementarea cu **MVar** si in implementarea cu **STM** (cursul 7).

5. Folosind implementarea **TMVar** si implementarea **Async** cu **TMVar** in **STM**, rescrieti exemplul de la punctul 3 c), folosind functia **waitAll** din **STM** care se executa atomic.

6. a) Rulati problema filozofilor in variantele din cursul 7 (definirea furculitelor cu **TVar Bool**, cu **TMVar Int**, varianta in care un filozof mananca de **n** ori).

   b) Modificati programul astfel incat sa va asigurati ca toate thread-urile filozof isi termina activitatea folosind **waitAll.**

   c) Considerand numai 2 filozofi (si numai 2 furculite) care mananca de n si, respectiv, de m ori, completati programul astfel incat sa afisati cine termina primul folosind **waitEither**. Pentru a folosi tipul de date **Either** threbuie sa importati **Data.Either**. Puteti verifica daca rezultatul este **Left** sau **Right** folosind functia **isLeft.**

   d) Modificati programul de la punctul b) astfel incat sa va asigurati ca unul dintre filozofi isi termina activitatea folosind **waitAny** si anuntati cine este acest filozof.

   Atentie: la c) si d) puteti intoarce valoarea solicitata (numele filozofului) ca rezultat al activitatii desfasurate asincron.