Računarske mreže 4R, Ispit - JUN2

30.06.2020.

Pročitati sve zadatke **pažljivo** pre rada - sve što nije navedeno ne mora da se implementira!
Na `Desktop`-u se nalazi zip arhiva. Unutar arhive se nalazi direktorijum u formatu `rm_rok_Ime_Prezime_miGGXXX` u kome se nalazi validan IntelliJ projekat. Izvući direktorijum iz arhive na Desktop i ubaciti svoje podatke u ime. Otvoriti IntelliJ IDEA, izabrati opciju `Open project` (ne `Import project`!) i otvoriti pomenuti direktorijum. Sve kodove ostaviti unutar već kreiranih Java fajlova. **Kodovi koji se ne prevode se neće pregledati. Nepoštovanje formata ulaza/izlaza nosi kaznu od -10% poena na zadatku!** Vreme za rad: **3h**.

1. **Operacije sa matricama (15p) (za studente koji nisu radili projekat)**
   Napraviti Java aplikaciju koja pomoću niti računa trag i transponant matrice.

   - Kao ulaz u program se daje putanja do tekstualnog fajla u kome se nalaze celobrojne matrice. U jednom redu se nalaze elementi jedne vrste matrice, razdvojeni blanko karakterom dok su matrice u fajlu su razdvojene novim redom (kao u primeru ispod). Učitati sadržaj fajla i ispisati matrice na standardni izlaz. (3p)
   - Označimo dimenziju matrice sa $n \times m$. Za transponovanje matrice, pokrenuti $n * m$ niti, pri čemu je svaka nit zadužena da promeni poziciju samo jednom elementu matrice. (5p)
   - Za računanje traga matrice, pokrenuti $min(n, m)$ niti, tako da svaka nit dodaje na ukupan zbir odgovarajući element sa glavne dijagonale. (5p)
   - Ispisati transponovanu matricu i trag matrice na standardni izlaz (kao u primeru ispod). (1p)
   - Postarati se da program ispravno obradjuje specijalnim slučajeve (npr. ako datoteka ne postoji na datoj putanji) i ispravno zatvoriti sve korišćene resurse u slučaju izuzetka. (1p)

   Primer datoteke:

   ```
   1 2 3
   4 5 6

   1 2
   3 4

   1 2
   3 4
   5 6
   ```

   Primer izlaza:

   ```
   1 4
   2 5
   3 6
   trag = 6
   1 3
   2 4
   trag = 5
   1 3 5
   2 4 6
   trag = 5
   ```

Napomena: Ohrabrujemo studente da koriste ***netcat*** *kako bi testirali delimične implementacije i otkrili greške pre vremena. Takodje, ukoliko se npr. preskoči implementacija servera, može se mock-ovati server putem* ***netcat****-a.*

Okrenite stranu!

2. **UDP (20p/12p)**
Napraviti klijent-server aplikaciju preko koje se dobija binarni zapis brojeva.

- Napisati Java klasu koja ima ulogu lokalnog UDP servera koji osluškuje na portu 10101 koristeći Java Datagram API. Server prihvata pakete od klijenata (maksimalne veličine `4B`). Za svaki prihvaćen paket ispisati poruku na standardni izlaz sa informacijala o rednom broju primljenog paketa i adresom sa koje je stigao paket. (6p/4p)
- Napisati Java klasu koja ima ulogu UDP klijenta koristeći Java Datagram API. Klijent sa standardnog ulaza učitava neoznačene cele brojeve, sve dok ne unese *KRAJ*. Za svaki učitan broj poslati serveru paket sa brojem kao sadržajem. Za svaki poslati paket, klijent dobija paket nazad od servera, čiji sadržaj ispisuje na standardni izlaz. (6p/4p)
- Za svaki prihvaćen paket, server klijentu koji je poslao paket šalje binarnu reprezentaciju celog broja izvučenog iz sadržaja paketa. (6p/3p)
- Postarati se da su svi resursi ispravno zatvoreni u slučaju izuzetka. (2p/1p)

3. **NonBlockingIO (25p/18p)**
Napraviti client-server TCP aplikaciju preko koje se računaju zajednički troškovi cimera.

- Napisati Java klasu koja ima ulogu lokalnog **neblokirajućeg** TCP servera, koji osluškuje na portu 12345 koristeći Java Channels API. (4p/3p)
- Napisati Java klasu koja ima ulogu **blokirajućeg** TCP klijenta koristeći Java Channels API. Klijent formira konekciju sa lokanim serverom na portu 12345 a zatim šalje serveru nisku učitanu sa standardnog ulaza koja predstavlja ime cimera, Posle poslatog imena klijent učitava instrukcije sa standardnog ulaza (po jednu u svakoj liniji) i izvršava operacije na osnovu tipa instrukcije: (7p/5p)
  - Ako učitana linija predstavlja pozitivan broj, taj broj predstavlja količinu novca koju je klijent uložio u zajednički život sa svojim cimerima. Za takav ulaz, klijent šalje serveru taj broj.
  - Ako je učitana niska `stanje`, šalje se serveru tu i klijent očekuje odgovor od servera, koji ispisuje na standardni izlaz.
  - Ako je učitana niska `kraj`, klijent raskida vezu i prestaje sa radom.
  - Ako tip instrukcije ne odgovara navedenim tipovima, ignorisati liniju i nastaviti sa radom.
- Server vodi evidenciju o uloženom novcu za svakog cimera. Server vrši operacije na osnovu tipa primljenog sadržaja od klijenata: (12p/9p)
  - Ako je primljen pozitivan ceo broj, server dodaje taj broj na ukupnu količinu novca koji je odgovarajući cimer uložio.
  - Ako je primljen zahtev `stanje`, server šalje klijentu ime svakog cimera i količinu novca koji je taj cimer uložio.
- Postarati se da su svi resursi ispravno zatvoreni u slučaju izuzetka. (2p/1p)