Računarske mreže, Ispit - JAN2 2023

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_mXGGXXX` 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!**

1. **PGN (20p)**
   Šahovske igre se najčešće čuvaju u standardizovanom formatu koji se naziva *Portable Game Notation*, ili skraćeno PGN. Za potrebe ovog zadatka, pretpostavićemo da svaki PGN fajl sadrži podatke o tačno jednoj igri. Vaš cilj je da parsirate ove fajlove i na standardni izlaz ispišete neke podatke. Struktura svakog fajla je sledeća:

   (a) Svi redovi do prvog praznog reda čine zaglavlje fajla, gde su dati podaci o tome ko je igrao, kada i gde je partija odigrana i sl. Postoji najmanje 7 ovakvih linija, ali ih može biti i više.
   (b) Zatim, sledi prazan red.
   (c) Svi ostali redovi predstavljaju podatke o potezima i oznaku rezultata. Nije potrebno parsirati ili posebno obrađivati ovaj deo fajla.

   Recimo, fajl koji predstavlja igru koja je odigrana danas između igrača Igrac Beli i Igrac Crni i koja je završena u 4 poteza bi mogao izgledati ovako:

```
[Event "RM Ispit"]
[Site "Matematicki fakultet, Beograd"]
[Date "2023.02.06"]
[Round "?"]
[White "Beli, Igrac"]
[Black "Crni, Igrac"]
[Result "1-0"]

1. e4 e5 2. Qh5 Nc6 3. Bc4 Nf6 4. Qxf7#  1-0
```

   Pretpostaviti da su svi fajlovi ispravno formatirani.

   - Sa standardnog ulaza se unosi naziv direktorijuma i godina, u zasebnim redovima. Godina će uvek biti četvorocifreni ceo broj. Obići sve fajlove u ovom direktorijumu i u njegovim poddirektorijumima i za svaku datoteku koji se završava sa `.pgn` pokrenuti zasebnu nit koja će obrađivati fajl. (5p)
   - Odštampati zaglavlja svih igara odigranih godine koja je uneta sa standardnog ulaza. Obratiti pažnju da zaglavlja neće uvek biti iste dužine! Datum će se uvek nalaziti u zaglavlju koje počinje sa ,,[Date" i biće u formatu `[Date "godina.mesec.datum"]` gde će ,,godina", ,,mesec" i ,,datum" biti ili brojevi ili znakovi pitanja (,,?") ako tačan datum nije poznat. Između zaglavlja različitih igara, ispisati prazan red. (5p)
   - Postarati se da se ispisi iz različitih niti ne prepliću. (4p)
   - Na kraju, ispisati broj igara koje su odigrane te godine. Da biste dobili poene za ovu stavku, neophodno je da se vaš program izvršava u više niti. (3p)
   - Postarati se da program obrađuje sve slučajeve, ispravno zatvara sve resurse i uspešno se završava. (3p)

   Primere ulaza i izlaza možete naći u direktorijumu `1-test-primeri`. Ulaz i izlaz svakog test primera se nalaze u zasebnim fajlovima, tako da je u fajlu `1.in` ulaz za prvi test primer, a u fajlu `1.out` odgovarajući izlaz, u fajlovima `2.in` i `2.out` su ulaz i izlaz za drugi, i tako dalje. Redosled odštampanih zaglavlja u izlazu nije bitan (ne mora da se poklapa sa redosledom koji je dat u izlaznim fajlovima).

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. **TCP Sockets (15p)**
Implementirati server, koji će imati ulogu da održava *in-memory* tabelu šahista i njihove trenutne rejtinge. Tabela ima kolone: `id (int)`, `naziv (String)` i `elo (int)`.

- Napraviti Java aplikaciju koja ima ulogu klijenta. Povezati se na lokalni server na portu 1996 koristeći **Socket** klasu. Nakon formiranja konekcije, klijent može poslati više zahteva serveru (zahtevi se unose sa standardnog ulaza), sve dok mu ne pošalje `bye`. Odgovori servera na zahtev se ispisuju na standardni izlaz. Mogući zahtevi su (implementirati u ovoj stavci samo slanje od strane klijenta): (3p)
  - *sel id* (`id` je tipa `int`)
  - *ins naziv* (`naziv` je tipa `String`)
  - *upd id elo* (`id` i `elo` su tipa `int`)
- Napraviti Java aplikaciju koja ima ulogu servera. Pokrenuti lokalni server na portu 1996, koristeći **ServerSocket** klasu. Server za svakog primljenog klijenta pokreće zasebnu nit u kojoj će se taj klijent obraditi tako što se ispiše poruka o pristiglom klijentu kao u primeru ispod. (2p)
- Server pristigle zahteve obradjuje na sledeći način:
  - *sel id*: vraća naziv i elo šahiste sa datim identifikatorom `id` (2p)
  - *ins naziv*: ubacuje u tabelu šahistu sa datim imenom dodeljujući mu jedinstveni identifikator (sledeći slobodan ceo broj) i elo u vrednosti 1300 (to je najmanja moguća vrednost za elo) i vraća poruku o uspešnosti operacije (2p)
  - *upd id deltae*: vrši izmenu elo vrednosti šahiste sa identifikatorom `id` za `deltae` i vraća poruku o uspešnosti operacije (2p)
- Ukoliko bilo koji od ovih zahteva nije ispravno formiran ili nije naveden iznad, vratiti tekst kao u primerima ispod. (1p)
- Imajte u vidu da mogu da se dese konfliktne situacije (kao npr. da dva klijenta žele da promene elo istoj osobi). Obezbediti da se ovakvi zahtevi pravilno obrade. Takođe, obezbediti da u slučaju izuzetaka, resursi budu ispravno zatvoreni. (3p)

```
> ins Magnus Carlsen
ins je uspesno izvrsen
> sel 1
Magnus Carlsen: 1300
> upd 1 30
upd je uspesno izvrsen
> sel 1
Magnus Carlsen: 1330
> upd 1 -10
upd je uspesno izvrsen
> sel 1
Magnus Carlsen: 1320
> bye
```

```
> ins Fabiano Caruana
ins je uspesno izvrsen
> sel 1
Fabiano Caruana: 1300
> upd 1 1500
upd je uspesno izvrsen
> sel 1
Fabiano Caruana: 2800
> upd 1 -10000
upd je uspesno izvrsen
> sel 1
Fabiano Caruana: 1300
> bye
```

```
> ins Marko
ins je uspesno izvrsen
> sel 1
Marko: 1300
// drugi klijent: upd
> sel 1
Marko: 1400
> ins Marko
ins je uspesno izvrsen
// nije isti Marko
> sel 2
Marko: 1300
> bye
```

---

*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!

3. **MinPrice (20p)**

Implementirati klijent-server aplikaciju koja koristi *Java Datagram API*.

U datoteci `hist-5cd7e29c.txt` date su dve kolone sa informacijama o kretanju cena artikla čiji je *ID* naveden u imenu datoteke. Podaci su automatski preuzimani sa sajtova više različitih prodavnica – centralizovani server obavljao je taj posao periodično, odbacujući najstarije podatke ukoliko veličina datoteke prevaziđe `16MB`.

S obzirom na povremene tehničke poteškoće poput pada mreže ili nedostupnosti sajta prodavnice, moguće je da za određene trenutke podaci nisu prikupljeni. U prvoj koloni datoteke nalazi se trenutak prikupljanja informacije, u formatu `dd.MM.yyyy,HH:mm`. Garantovano je da vrednosti u datoteci poštuju navedeni format i da su poređane u rastućem poretku. Druga kolona predstavlja minimalnu cenu artikla za dati trenutak, ili -1 ukoliko ni u jednoj prodavnici artikla nije bilo na stanju.

- Napisati program koji ima ulogu lokalnog *UDP* servera koji osluškuje na portu *12345*. Sa serverske strane potrebno je iz datagrama koji pristižu od klijenata pročitati 8 bajtova podataka kao celobrojnu vrednost `long t`. Učitani broj `t` predstavlja broj sekundi od *UNIX* epohe. Za svaki datagram potrebno je odgovoriti pošiljaocu svojim datagramom koji sadrži minimalnu cenu artikla u tom trenutku (što odgovara najažurnijoj vrednosti iz druge kolone do trenutka `t`). U slučaju da je `t` negativno ili ako odgovara vrednosti koja prethodi najstarijem datumu iz datoteke, klijentu poslati najstariju cenu. Odmah po slanju server ispisuje datum i vreme koji odgovaraju trenutku `t`, kao i samu cenu. (10p)
- Implementirati klijentsku stranu kao program nezavisan od servera. Na početku izvršavanja, sa standardnog ulaza se učitava datum u formatu `dd.MM.yyyy,HH:mm`. Ukoliko dati format nije ispoštovan, obustaviti rad klijenta sa ispisom poruke na `stderr`. Zatim se u okviru datagrama serveru šalje 8 bajtova koji odgovaraju `long t` vrednosti, koja predstavlja broj sekundi od *UNIX* epohe. Klijent treba da čeka na odgovor od servera ne duže od *5s*, a ukoliko odgovor ne stigne, da ponovi slanje istog zahteva. Ukoliko ni tada (nakon 5s) ne stigne odgovor, klijent treba da bude zaustavljen sa ispisom poruke o grešci. Kada odgovor pristigne, potrebno je izvršiti ispis pristiglih informacija. (9p)
- Postarati se da aplikacija vrši obradu grešaka i da ispravno zatvara resurse. (1p)

Napomena: Implementacija tipa podataka koji odgovara jednom redu datoteke `hist-5cd7e29c.txt`, kao i implementacija učitavanja i pretrage, data je u `DataPoints.java`. U nastavku je prikazan jedan primer interakcije sa programom, a dodatne primere ulaza i očekivanih izlaza možete naći u direktorijumu `3-test-primeri`.

```
// hist-5cd7e29c.txt:

01.01.2023,09:03 21999
01.01.2023,21:03 21999
02.01.2023,09:02 23105
...
13.01.2023,09:02 19999
13.01.2023,21:05 19999
14.01.2023,09:03 22050
14.01.2023,21:04 22050
```

```
// klijentska strana:

> 01.01.2023,12:00
Sending request...
01.01.2023,12:00 cost=21999

> 02.01.2023,06:00
Sending request...
02.01.2023,06:00 cost=21999

> 02.01.2023,10:00
Sending request...
02.01.2023,10:00 cost=23105

>14.01.2023,09:03
Sending request...
14.01.2023,09:03 cost=22050

>20.01.2023,12:00
Sending request...
20.01.2023,12:00 cost=22050
```

```
// serverska strana:

01.01.2023,12:00 cost=21999
02.01.2023,06:00 cost=21999
02.01.2023,10:00 cost=23105
14.01.2023,09:03 cost=22050
20.01.2023,12:00 cost=22050
```