Računarske mreže, Ispit - SEP1 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. **Log datoteke (20p)**
   Neko je napravio neočekivano veliku količinu zahteva ka veb serveru. Srećom, postoje datoteke sa informacijama o svakom zahtevu, kao i spisak adresa za koje znate da nisu sumnjive. Vaš zadatak je da izdvojite sa kojih IP adresa su došli potencijalno sumnjivi zahtevi.

   - Sa standardnog ulaza se unose dve niske. Prva je lokacija do datoteke gde se nalazi spisak opsega adresa koje nisu sumnjive (tzv. *whitelist*), svaka u jednom redu, koju je potrebno učitati. Druga je putanja do direktorijuma gde se nalaze log datoteke. Obići direktorijum sa logovima rekurzivno, uključujući sve njegove poddirektorijume. (5p)
   - Za svaku datoteku čiji naziv počinje sa `access_log` pokrenuti zasebnu nit koja je obrađuje. Svaki red u datoteci predstavlja jedan zahtev i počinje IPv4 adresom koju je potrebno izdvojiti. (3p)
   - Opsezi IP adresa u *whitelist* datoteci su dati u CIDR formatu, tj. oni su oblika 123.123.123.123/10, gde deo pre povlake predstavlja IP adresu, a deo posle prefiks bitova koji treba da se poklapa da bi se reklo da neka adresa pripada tom opsegu. Recimo, opseg 10.0.0.0/8 obuhvata sve adrese oblika 10.x.x.x, jer će prvih osam bitova u ovakvim adresama uvek biti 00001010 (binarno 10). S druge strane, opseg 108.162.192.0/18 obuhvata sve adrese između 108.162.192.0 i 108.162.255.255, ali ne npr. 108.162.191.15. Ispisati sve IP adrese iz log datoteka koje **ne** pripadaju ni jednom od opsega zadatih u *whitelist* datoteci. Parcijalno će biti bodovana rešenja koja daju ispravan rezultat samo za prefikse 8, 16, 24 i 32. (5p)
   - Obezbediti da se svaka IP adresa ispisuje samo jedanput i da nema preplitanja pri ispisu. (5p)
   - Postarati se da program obrađuje sve slučajeve, ispravno zatvara sve resurse i uspešno se završava. (2p)

   Primer *whitelist* i *log* datoteka se nalaze u direktorijumu `tests/zad1`, a očekivani izlaz u datoteci `tests/zad1/1.out`.

2. **Non-Blocking IO - Igra saberenih pogodaka (25p)**
Napraviti klijent-server Java aplikaciju koristeći *Channels API* preko koje će se igrati igra saberenih pogodaka.

- Napisati Java klasu koja ima ulogu **blokirajućeg** TCP klijenta, korišćenjem Java Channels API. Klijent formira konekciju sa serverom na portu 12345 i zatim šalje serveru kombinaciju od 5 celih brojeva, pri čemu je svaki iz intervala [-10, 10], učitanih sa standardnog ulaza. (8p)
- Napisati Java klasu koja ima ulogu **neblokirajućeg** TCP servera, korišćenjem Java Channels API, koji osluškuje na portu 12345. Server slučajnim izborom generiše kombinaciju od 5 celih brojeva, pri čemu je svaki iz intervala [-10, 10]. Nakon upostavljanja konekcije, klijent šalje serveru svoj pokušaj. Server izračuna koliko je poena ostvario klijent, i kao odgovor mu vrati broj ostvarenih poena. Server na sledeći način računa broj poena klijenta:

Server-ova kombinacija:	5	-5	7	0	-9
Klijent-ov pokušaj:	3	8	-4	-5	-9

Broj poena ostvarenih:
(5 - 3) * 5 + (-5 - 8) * (-5) + (7 - (-4)) * 7 + (0 - (-5)) * 0 + (-9 - (-9)) * (-9) = 152
Tj. suma za svaku od cifra u kombinaciji (vrednost servera - vrednost klijenta) * (vrednost servera) (15p)
- Postarati se da su svi resursi ispravno zatvoreni u slučaju izuzetka. (2p)

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. **UDP circles (15p)**
Implementirati klijent-server aplikaciju koja koristi *Java Datagram API*.

- Napisati program koji ima ulogu lokalnog *UDP* servera koji osluškuje na portu *12345*. (3p)
- Implementirati klijentsku stranu kao program nezavisan od servera. Na početku izvršavanja klijenta, sa standardnog ulaza u prvom redu se učitavaju podaci koji opisuju krug A. U pitanju su celi brojevi x_A, y_A i r_A kojima su predstavljene koordinate centra kruga (x_A, y_A) i poluprečnik kruga r_A. U narednom redu se zatim učitavaju takvi podaci za krug B. Pomoću datagrama serveru poslati obe učitane linije. (3p)
- Nakon što pošalje datagram, 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. (2p)
- Sa serverske strane se postarati da su primljeni podaci u odgovarajućem formatu. Ukoliko to nije slučaj, baciti izuzetak `RuntimeException` i poslati klijentu -1. U slučaju dobrog formata, server treba da uzvrati klijentu poruku da li je jedna kružnica u okviru druge, da li se samo dodiruju ili imaju presečne tačke. (4p)
- Na klijentskoj strani ispisati odgovor dobijen od servera. U slučaju da je primljeni datagram -1, ponoviti ceo postupak, počevši od učitavanja podataka za krugove A i B. (2p)
- Postarati se da se sve greške pravilno obrađuju i da se svi resursi zatvaraju. (1p)

```
// primer kada se A nalazi u B:

cl1-in> 1 -10 3
cl1-in> 4 4 20
cl1-out< Slanje podataka...
srv-out< Klijentska poruka je protumacena!
srv-out< Slanje podataka...
cl1-out< Server je odgovorio u roku od 5s!
cl1-out< Krug A se nalazi u krugu B.

// primer kada od servera ne stizu poruke:

cl2-in> 1 -10 3
cl2-in> 4 4 20
cl2-out> Slanje podataka...
cl2-out> Server nije odgovorio u roku od 5s!
cl2-out> Slanje podataka ponovo...
cl2-out> Server nije odgovorio u roku od 5s!
cl2-out> Obustavljanje rada klijenta!

// primer kada nema preseka kruznica:

cl3-in> 3 -9 7
cl3-in> -7 -15 3
cl3-out> Slanje podataka...
srv-out> Klijentska poruka je protumacena!
srv-out> Slanje podataka...
cl3-out> Server je odgovorio u roku od 5s!
cl3-out> Kruznice A i B nemaju presek.
```

```
// primer kada se najpre procitaju
// podaci u neispravnom formatu

cl4-in> (1, 4) 1
cl4-in> (-1, 4) 1
cl4-out> Slanje podataka...
cl4-out> Server je odgovorio u roku od 5s!
srv-out> Poruka ima los format!
srv-out> Obavestavanje klijenta o tome...
cl4-out> Poruka ima los format!
cl4-in> 1 4 1
cl4-in> -1 4 1
cl4-out> Slanje podataka...
srv-out> Klijentska poruka je protumacena!
srv-out> Slanje podataka...
cl4-out> Server je odgovorio u roku od 5s!
cl4-out> Kruznice A i B se dodiruju.

// primer kada se kruznice seku:

cl5-in> 0 5 4
cl5-in> -1 2 1
cl5-out> Slanje podataka...
srv-out> Klijentska poruka je protumacena!
srv-out> Slanje podataka...
cl5-out> Server je odgovorio u roku od 5s!
cl5-out> Kruznice A i B se presecaju.
```