# Mreže računala

Prvi kolokvij, 26. studenoga 2025. godine
Rješenja programerskih zadataka

## DRUGI ZADATAK

UKUPNO BODOVA 13

Dovršite ispod prikazani kod servera. Server komunicira s klijentima na IP-adresi 161.53.8.14 i portu 65100. Za svakog klijenta za koji server prihvati konekciju, server određuje broj klijentovih alternativnih *host-nameova* te šalje klijentu taj broj. Nakon toga, server klijentu šalje koji je on ukupno po redu klijent kojem je prihvaćena konekcija. Primjerice, ako je neki klijent 2025. po redu klijent s kojim server komunicira (brojeći od trenutka pokretanja servera), tada tom klijentu treba poslati broj 2025. Zatim server prekida konekciju s klijentom.

Server ne treba moći istovremeno komunicirati s više klijenata, a najveći broj klijenata koji odjednom može čekati na uslugu servera je 5. Pripazite na moguće greške (u slučaju greške ispišite prikladnu poruku o grešci). Možete pretpostaviti da se svaki broj može poslati jednom *send* naredbom.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <malloc.h>

#include <sys/socket.h>
#include <sys/types.h>
#include <netdb.h>
#include <netinet/in.h>
#include <unistd.h>
#include <arpa/inet.h>

int main(void) {
    int listenerSocket = socket(PF_INET, SOCK_STREAM, 0);
    if(listenerSocket == -1)
        perror("socket");

    // ---------- RJEŠENJE ----------

    // ---------- bind dio ---------- (3 boda)

    struct sockaddr_in mojaAdresa;
    mojaAdresa.sin_family = AF_INET;
    mojaAdresa.sin_port = htons(65100);
    mojaAdresa.sin_addr.s_addr = INADDR_ANY;
    memset(mojaAdresa.sin_zero, '\0', 8);
    if(bind(listenerSocket, (struct sockaddr *)&mojaAdresa,
            sizeof(mojaAdresa)) == -1)
        perror("bind");
```

```
    // ---------- listen dio ---------- (1 bod)

    if(listen(listenerSocket, 5) == -1)
        perror("listen");


    int brojac = 0;
    while(1) {

        // ---------- accept dio i ažuriranje brojača ---------- (2 boda)

        struct sockaddr_in klijentAdresa;
        int lenAddr = sizeof(klijentAdresa);
        int commSocket = accept(listenerSocket,
                        (struct sockaddr *)&klijentAdresa, &lenAddr);
        if(commSocket == -1)
            perror("accept");
        ++brojac;

        // ------ određivanje broja alternativnih adresa ------ (3 boda)
        struct hostent *hostInfo;
        hostInfo = gethostbyaddr((const char *)&(klijentAdresa.sin_addr),
                                 sizeof(klijentAdresa.sin_addr), AF_INET);
        if(hostInfo == NULL)
            herror("gethostbyaddr");
        int brAlt = 0;
        while(hostInfo->h_aliases[brAlt] != NULL)
            ++brAlt;

        // ------ slanje broja alternativnih adresa ------ (1.5 bodova)

        brAlt = htonl(brAlt);
        if(send(commSock, &brAlt, sizeof(brAlt), 0) != sizeof(brAlt))
            perror("send");

        // ------ slanje vrijednosti brojača ------ (1.5 bodova)

        int nbrojac = htonl(brojac);
        if(send(commSock, &nbrojac, sizeof(nbrojac), 0) != sizeof(nbrojac))
            perror("send");

        // ------ zatvaranje utičnice za klijenta ------ (1 bod)

        close(commSocket);
    }

    return 0;
}
```

## TREĆI ZADATAK

OSTVARENI BODOVI | | 12 |

U ovom zadatku se **ne očekuje** pisanje programskog koda.

Potrebno je dizajnirati mrežnu aplikaciju koja omogućava igranje igre *Kviz*. Svaki igrač prije početka mora odabrati svoje ime od točno 8 znakova. Ime svakog igrača mora biti jedinstveno. Igrači koji su u igri podijeljeni su u 5 soba od kojih svaka ima svoju temu: *matematika*, *geografija*, *fizika*, *kemija*, *biologija*. Svaki igrač mora biti u nekoj sobi koju odabere, ali ju u bilo kojem trenutku može promijeniti. Svaki igrač koji se nalazi u pojedinoj sobi istovremeno dobiva isto pitanje vezano uz temu te sobe (primjerice, za sobu s temom *biologija* pitanje *Koliko pauk ima nogu?*). Samo igrač koji prvi da točan odgovor na pitanje ostvaruje jedan bod. Nakon toga svi igrači u toj sobi moraju vidjeti koji je bio točan odgovor i tko je dao taj odgovor. Igrač u bilo kojem trenutku može napustiti igru ili vidjeti koliko ima bodova i na kojem je mjestu u rang ljestvici svih trenutno aktivnih igrača po broju bodova.

(a) Osmislite vrste poruka i njihov format (zaglavlje/*header* i tijelo/*payload*) koje razmjenjuju klijent i server. 6 bodova

(b) Navedite kakvi se sve tipovi grešaka mogu dogoditi u komunikaciji. 2 boda

(c) Odaberite primjer po jedne klijentske poruke za svaku vrstu poruke klijenta (primjeri u skladu s Vašim definicijama mogućih vrsta i formata poruka) te pripadne odgovore od strane servera. 4 boda

**Rješenje**. Prikazan je jedan od mogućih dizajna sustava komunikacije.

(a) Svaka poruka sastoji se od 2 dijela: zaglavlja i tijela poruke. Zaglavlje se sastoji od dva integera: prvi predstavlja duljinu poruke (ne uključujući zaglavlje), a drugi predstavlja kod kojim se jednoznačno određuje vrsta poruke.

Vrste poruka između servera i klijenta su:

(1) *IGRAJ ime soba* - klijent šalje serveru svoje ime s kojim želi igrati i sobu koju je odabrao[1]

(2) *PROMIJENI soba* - klijent šalje serveru poruku da želi promijeniti sobu u *soba*

(3) *PITAJ pitanje* - server šalje klijentu pitanje

(4) *ODGOVORI odgovor* - klijent šalje serveru odgovor na postavljeno pitanje

(5) *ODGOVORI_R jel_tocno* - server šalje klijentu povratnu informaciju je li njegov ponuđeni odgovor točan ili ne (string ”DA” ili ”NE”)

(6) *RJESENJE odgovor ime_igraca* - server šalje klijentu točan odgovor na posljednje pitanje postavljeno u sobi i ime igrača koji je dao točan odgovor

(7) *INFO* - klijent želi od servera saznati koliko ima bodova i na kojem je mjestu u rang ljestvici

(8) *INFO_R bodovi mjesto* - server šalje klijentu poruku s njegovim brojem bodova i mjestom na rang ljestvici

[1] Ovdje su se mogle staviti dvije vrste poruke. Jedna vrsta poruke u kojoj klijent serveru javlja ime igrača te druga u kojoj javlja u kojoj prostoriji igrač želi biti. Obzirom da prema zadatku svaki igrač mora biti u nekoj prostoriji, ne može postojati igrač koji nije niti u jednom prostoriji u nekom trenutku pa može i ovako.

(9) *BOK* - klijent javlja serveru da želi napustiti igru. Nakon toga server prekida komunikaciju.

(10) server na **svaki** klijentov zahtjev odgovara je li zahtjev uspio ili nije - odgovor je oblika: *ODGOVOR poruka* gdje je poruka string koji je jednak ”OK” ako je zahtjev uspješno obrađen, a inače sadrži opis greške.

(b) Tipovi grešaka u komunikaciji:

- klijent je odabrao neispravno ime igrača ili ime koje već ima neki aktivni igrač (jer prema zadatku, ime svakog igrača mora biti jedinstveno)
- klijent je odabrao nepostojeću sobu

(c) Navodimo po jedan primjer klijentske poruke za svaku vrstu poruke **klijenta** te zatim pripadne odgovore od strane servera:

(1) klijent: [19,IGRAJ] kvizas00 matematika
server: [2,ODGOVOR] OK

(2) klijent: [12,PROMIJENI] astronautika
server: [19,ODGOVOR] Ne postoji ta soba.

(4) klijent: [11,ODGOVORI] realan broj
server: [2,ODGOVOR] OK
server: [2,ODGOVORI_R] NE
server: [22,RJESENJE] prirodan broj stranger
(Napomena: Gornju poruku tipa RJESENJE server neće poslati ako nitko još nije odgovorio točno na postavljeno pitanje.)

(7) klijent: [0,INFO]
server: [2,ODGOVOR] OK
server: [6,INFO_R] 100 50

(9) klijent: [0,BOK]
server: [2,ODGOVOR] OK

## ČETVRTI ZADATAK

OSTVARENI BODOVI | | 17 |

Tvrtka *Instrukcije Things* na kraju svakog mjeseca isplaćuje svojim zaposlenicima plaću prema broju sati instrukcija koje su taj mjesec odradili. U ovome zadatku promatramo mrežnu aplikaciju kojom zaposlenici nakon svake odrađene instrukcije prijavljuju koliko sati su te instrukcije trajale (kako bi se na kraju mjeseca odredio njihov ukupan broj). Svakom zaposleniku prilikom zaposlenja dodijeljen je njegov jedinstven prirodan broj (strogo) manji od 1000 koji ga identificira. Osim tog broja, pamti se njegovo ime, prezime i područja za koje daje instrukcije, a svaki od tih podataka je string duljine najviše 20. Područja su: matematika, fizika, kemija, informatika i geografija. Svaki zaposlenik može davati instrukcije iz više područja. U komunikaciji, zaposlenik osim svog identifikatora koristi i svoju šifru kako bi potvrdio svoj identitet. Šifra se sastoji od točno 8 znakova i ne sadrži znak točku (`'.'`). Broj sati u ovome zadatku je tipa `int`.

Protokol za komunikaciju serverskog i klijentskog programa definiran je na sljedeći način:

- Zaglavlje poruke sastoji se od koda (koji određuje vrstu poruke) te jedinstvenog broja koji određuje zaposlenika.
- Vrste poruka uključuju sljedeće (u drugom stupcu je naveden kod; uočite da klijent u svojim porukama potvrđuje svoj identitet svojom lozinkom *lozinka*):

| | | |
|---|---|---|
| *DODAJ lozinka br_sati* | 1 | Korisnik želi evidentirati broj sati instrukcija koje je odradio. |
| *INFO lozinka* | 2 | Korisnik želi saznati koliko je do sad ukupno odradio sati. |
| *INFO_R br_sati* | 3 | Server šalje klijentu broj sati koje je on do sad ukupno odradio. |
| *ODGOVOR poruka* | 4 | Server šalje string koji je jednak "OK" za uspješno obrađen zahtjev ili opis greške. |

- Pretpostavimo da su implementirane funkcije za slanje i primanje poruke (s utičnice `sock`):
  - `int posalji(int sock, const char *poruka)`
  - `int primi(int sock, char **poruka)`

  pri čemu u tim funkcijama `poruka` predstavlja cijelu poruku (zaglavlje + tijelo) gdje su zaglavlje i tijelo, kao i komponente zaglavlja, odijeljeni znakom zareza (`','`). Sami odredite kako su odijeljene komponente tijela poruke. Funkcije vraćaju 0 ako je došlo do greške (inače 1).

(1.) Implementirajte **sve** strukture/polja/varijable koje trebaju serveru za pamćenje svih navedenih podataka o zaposlenicima. 4 boda

(2.) Implementirajte serversku funkciju `void evidentiraj(int sock, char *poruka)` (oprez: `poruka` kao i gore ima zaglavlje + tijelo) koja ažurira odgovarajuća polja i strukture na serveru te vraća poruku klijentu o uspješnom ili neuspješnom (u kom slučaju poruka sadrži opis greške) evidentiranju broja sati održane instrukcije. Evidentiranje nije uspjelo ako vrijedi barem jedno od sljedećeg: ne postoji korisnik s jedinstvenim brojem iz zaglavlja, lozinka tog korisnika nije ispravna, broj sati je negativan ili veći od 8 (tvrtka ne dopušta da jedne instrukcije traju više od 8 sati). 7 bodova

(3.) Implementirajte klijentsku funkciju `void informacije(int sock)` koja od korisnika učita njegov jedinstveni broj i lozinku. Zatim funkcija pita server koliko taj korisnik ima ukupan broj sati odrađenih instrukcija. Potrebno je korisniku ispisati podatke dobivene od servera (ili poruku o grešci ako se ona dogodila). 6 bodova

**Rješenje četvrtog zadatka**.

1. Jednostavno rješenje je uzeti polje od 1000 struktura - na indeksu `i` u tom polju nalaze se podaci za zaposlenika čiji je dodijeljeni jedinstveni prirodan broj jednak `i`. Možemo staviti, primjerice, da niti jedan zaposlenik nema broj `i` kao svoj jedinstveni broj ako je `zaposlenici[i].sifra` prazan string (tj. vrijedi `zaposlenici[i].sifra == '\0'`).

```
typedef struct {
      char ime[21], prezime[21];
      char (*podrucja)[21]; /* dinamički alociran (zadnji je NULL)[2] */
      char sifra[8];
      int ukupno_sati;
} zaposlenik;

zaposlenik zaposlenici[1000];
```

2. Komponente tijela poruke bit će odvojene znakom točke (`'.'`) obzirom da taj znak prema zadatku ne može biti dio šifre.

```
void evidentiraj(int sock, char* poruka) {
      /* izvadimo dijelove iz poruke klijenta */
      int kod, id, br_sati;
      char poruka[100]; /* 100 je dovoljno za poruke koje šaljemo */
      /* ne znamo ispravnost lozinke (8 znakova) pa alociramo dovoljno
         mjesta (lozinka je dio od poruka pa sigurno ima manju duljinu) */
      char* lozinka = (char*)malloc((strlen(poruka)+1)*sizeof(char));
      sscanf(poruka, "%d,%d,%[^.].%d", &kod, &id, lozinka, &br_sati);

      /* napravit ćemo samo one provjere koje se traže u zadatku */
      if(zaposlenici[id].sifra == '\0') {
            sprintf(poruka,"4,%d,Ne postoji taj zaposlenik.", id);
      } else if(strcmp(zaposlenik[id],lozinka) != 0) {
            sprintf(poruka,"4,%d,Lozinka nije ispravna.", id);
      } else if(br_sati < 0 || br_sati > 8) {
            sprintf(poruka,"4,%d,Broj sati nije dozvoljen.", id);
      } else {
            zaposlenici[id].ukupno_sati += br_sati;
            sprintf(poruka,"4,%d,OK", id);
      }
      posalji(sock, poruka);
}
```

3. 
```
void informacije(int sock) {
      /* učitamo tražene podatke od korisnika */
      int broj, i;
      char lozinka[9]; /* 8 znakova i '\0' */
      printf("Unesi svoj jedinstveni broj: "); /* opcionalan ispis */
```

[2] Alternativno, mogli smo staviti `char podrucja[5][21];` obzirom da prema zadatku znamo da najviše može biti 5 područja (ako su, primjerice, samo 3 za nekog zaposlenika, preostala 2 su prazni stringovi).

```
        scanf("%d", &broj);
        printf("Unesi tocno 8 znakova svoje sifre: ");
        for(i = 0; i < 8; ++i) {
              scanf("%c", &lozinka[i]);
        }
        lozinka[8] = '\0'; /* za kraj stringa */

        /* šaljemo serveru poruku s upitom */
        char poruka[100];
        sprintf(poruka, "2,%d,%s", broj, lozinka);
        int poslano = posalji(sock, poruka);
        if(poslano == 0) {
              printf("Greska pri slanju poruke!");
              return;
        }

        /* primamo odgovor servera */
        char* odgovor;
        int primljeno = primi(sock, &odgovor);
        if(primljeno == 0) {
              printf("Greska pri primanju poruke!");
              return;
        }

        /* izvadimo dijelove odgovora i ispišemo ih korisniku */
        int kod;
        char* ispis = (char*)malloc((strlen(odgovor)+1)*sizeof(char));
        sscanf(odgovor,"%d,%d,%[^\n]", &kod, &broj, &ispis);
        if(kod == 3) {
              printf("Ukupan broj sati: %s", ispis);
        } else if(kod == 4) {
              printf("Greska: %s", ispis);
        } else {
              printf("Neocekivana vrsta poruke!");
        }
}
```