Strukture podataka i algoritmi 1
(zadatak - max 30 poena)

Avgust, 2026

Sportski klub vodi evidenciju o učešću svojih članova u različitim takmičenjima. Za svakog člana u bazi već ima sledeće podatke: njegov ID u klubu (ceo broj), ime i prezime (niz karaktera) i godina učlanjenja (ceo broj). Prilikom prijave takmičenja koje klub organizuje ili na kome učestvuje unose se ID takmičenja (ceo broj), naziv takmičenja (niz karaktera), godina održavanja (ceo broj), broj prijavljenih članova i za svakog prijavljenog člana njegov ID. Jedan član može učestvovati na više takmičenja. Evidencija rezultata članova na takmičenjima se vodi odvojeno. Za svaku evidenciju pamte se ID takmičenja, ID člana i broj osvojenih poena. Na osnovu tih podataka ažurira se ukupan broj osvojenih poena za svakog člana na takmičenjima. Takmičenje se smatra završenim kada su za sve prijavljene članove evidentirani rezultati. Završena takmičenja se uklanjaju iz liste takmičenja.
Napisati program u kome se učtavaju podaci o članovima kluba, a zatim podaci o takmičenjima. Potom se vrši ažuriranja podataka o takmičerima na osnovi evidencije o učešću na takmičenju. Završena takmičenja se uklanjaju iz liste takmičenja. Na kraju ispisati listu takmičenja koja nisu završena i podatke o članu koji je učestvovao na najvećem broju takmičenja.

(kompletna main funkcija - 4 poena, obavezno)

Za rešavanje problema napisati sledeće funkcije:

a) Definisati sve potrebne složene tipove podataka neophodne za rešavanje opisanog problema.
(3 poena, obavezno)
b) Napisati funkciju **UcitajClanove** koja iz datoteke *Clanovi.txt* učitava podatke o članovima kluba i formira listu/niz članova.
(3 poena, obavezno)
c) Napisati funkciju **NadjiClana** koja za dati ID člana vraća pokazivač na člana, pri čemu postoji mogućnost da član sa datim ID-jem ne postoji.
(2 poena)
d) Napisati funkciju **UcitajTakmičenja** koja iz datoteke *Takmičenja.txt* učitava podatke o takmičenjima, formira listu/niz takmičenja i vrši raspoređivanje članova po takmičenjima.
(4 + 2 poena)
e) Napisati funkciju **Azuriranje** koja iz datoteke *Evidencija*.txt učitava evidencija o rezultatima članova kluba na takmičenju i ažurira podatke o članoviam kluba i takmičenjima.
(4 poena)
f) Napisati funkciju **ObrisiTakmicenje** koja uklanja završeno takmičenje iz liste/niza takmičenja.
(5 poena)
g) Napisati funkciju **Ispis** koja ispisuje člana kluba koji je učestvovao na najvećem broju takmičenja.
(3 poena)

Pri učitavanju podataka podrazumevati da su svi podaci korektno zadati.
Podaci iz datoteke se mogu učitavati uz pretpostavku da se na početku nalazi broj podataka u datoteci ili se podaci mogu učitavati do kraja datoteke.
Dozvoljeno je proširivanje struktura i definisanje novih, kao i definisanje drugih funkcija.
Maksimalan broj poena se dobija samo u slučaju da postoji veza između struktura koje čuvaju članove i struktura koje čuvaju takmičenja, tj. ako postoje pokazivači "na jednu" ili "na obe strane" i da u strukturama nema dupliranja podataka.
Optimalno definisanje struktura i funkcija koje se mogu koristiti više puta donosi bonus poene.

Zadatak se može rešavati korišćenjem povezanih lista, nizova ili kombinacijom.
Zadatak rešiti bez korišćenja globalnih promenljivih i bez unapred definisanih dužina korišćenih nizova (izuzetak može da bude pomoćni niz za učitavanje stringova)**.**
Za **funkcije koje nisu definisane** mora da postoji deklaracija funkcije i kratak opis koji deo problema funkcija treba da reši i mogu se pozivati kao da postoji kompletno definisana funkcija.

Rešenje studenta

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

#define LEN 256

typedef struct clan {
    int id;
    char *ime_prezime;
    int godinaUclanjenja;
    int ukupnoPoena;
    int br_takmicenja;
    struct clan *next;
} Clan;

typedef struct veza {
    Clan *clan;
    struct veza *next;
} Veza;

typedef struct takmicenje {
    int id;
    char *naziv;
    int godinaOdrzavanja;
    int brojPrijavljenih;
    int evidentirano; //pamti broj takmičara sa evidentiranim rezultatima
    Veza *lista_clanova;
    struct takmicenje *next;
} Takmicenje;


Clan *NadjiClana(Clan *glava, int id)
{
    while (glava != NULL) {
        if (glava->id == id)
            return glava;
        glava = glava->next;
    }
    return NULL;
}

Takmicenje *NadjiTakmicenje(Takmicenje *lista, int id)
{
    while (lista != NULL) {
        if (lista->id == id)
            return lista;
        lista = lista->next;
    }
    return NULL;
}

void OslobodiClanove(Clan *lista)
{
    while (lista != NULL) {
        Clan *tmp = lista;
        lista = lista->next;
        free(tmp->ime_prezime);
        free(tmp);
    }
}

void OslobodiVeze(Veza *lista)
{
```

```
    while (lista != NULL) {
        Veza *tmp = lista;
        lista = lista->next;
        free(tmp);
    }
}

void OslobodiTakmicenja(Takmicenje *lista)
{
    while (lista != NULL) {
        Takmicenje *tmp = lista;
        lista = lista->next;

        OslobodiVeze(tmp->lista_clanova);
        free(tmp->naziv);
        free(tmp);
    }
}

Clan *UcitajClanove(){

    FILE *f = fopen("Clanovi.txt", "r");
    if (f == NULL){
        printf("Greška pri otvaranju dokumenta\n");
        return NULL;
    }

    Clan *glava = NULL;
    char naziv[LEN];

    while (!feof(f)) {

        Clan *novi = (Clan*)malloc(sizeof(Clan));

        if (novi == NULL) {
            printf("Greška\n");
            fclose(f);
            return NULL;
        }

        if (fscanf(f, "%d %255s %d",
                   &novi->id,
                   naziv,
                   &novi->godinaUclanjenja) != 3) {
            free(novi);
            break;
        }

        novi->ime_prezime =(char*)malloc((strlen(naziv) + 1) * sizeof(char));
        strcpy(novi->ime_prezime, naziv);

        novi->ukupnoPoena = 0;
        novi->br_takmicenja = 0;

        novi->next = glava;
        glava = novi;
    }

    fclose(f);
    return glava;
}

Takmicenje *UcitajTakmicenja(Clan *clanovi)
{
    FILE *f = fopen("Takmicenja.txt", "r");
```

```
if (f == NULL){
    printf("Greška pri otvaranju dokumenta\n");
    return NULL;
}

Takmicenje *glava = NULL;
char str[LEN];

while (!feof(f)) {

    Takmicenje *novo = (Takmicenje*)malloc(sizeof(Takmicenje));

    if (novo == NULL) {
        printf("Greška\n");
        fclose(f);
        return NULL;
    }

    if (fscanf(f, "%d %255s %d %d",
               &novo->id,
               str,
               &novo->godinaOdrzavanja,
               &novo->brojPrijavljenih) != 4) {
        free(novo);
        break;
    }

    novo->naziv =(char*)malloc((strlen(str) + 1) * sizeof(char));
    strcpy(novo->naziv, str);

    novo->evidentirano = 0;
    novo->lista_clanova = NULL;

    for (int i = 0; i < novo->brojPrijavljenih; i++) {

        int idClana;

        if (fscanf(f, "%d", &idClana) != 1)
            break;

        Clan *clan = NadjiClana(clanovi, idClana);

        if (clan != NULL) {

            Veza *novaVeza = (Veza*)malloc(sizeof(Veza));

            if (novaVeza == NULL) {
                fclose(f);
                return NULL;
            }

            novaVeza->clan = clan;

            novaVeza->next = novo->lista_clanova;
            novo->lista_clanova = novaVeza;

            clan->br_takmicenja++;
        }
    }

    novo->next = glava;
    glava = novo;
}
```

```
    fclose(f);
    return glava;
}



void Azuriranje(Clan *clanovi, Takmicenje *takmicenja)
{
    FILE *f = fopen("Evidencija.txt", "r");
    int idTakmicenja, idClana, poeni;

    if (f == NULL)
        return;

    while (fscanf(f, "%d %d %d", &idTakmicenja, &idClana, &poeni) == 3) {

        Takmicenje *takmicenje = NadjiTakmicenje(takmicenja, idTakmicenja);

        Clan *clan = NadjiClana(clanovi, idClana);

        if (takmicenje == NULL || clan == NULL)
            continue;

        Veza *veza = takmicenje->lista_clanova;

        while (veza != NULL && veza->clan != clan)
            veza = veza->next;

        if (veza != NULL) {
            clan->ukupnoPoena += poeni;
            takmicenje->evidentirano++;
        }
    }

    fclose(f);
}


void ObrisiTakmicenje(Takmicenje **lista, int idTakmicenja)
{
    Takmicenje *tr = *lista;
    Takmicenje *preth = NULL;

    while (tr && tr->id != idTakmicenja) {
        preth = tr;
        tr = tr->next;
    }

    if (tr == NULL)
        return;

    if (preth == NULL)
        *lista = tr->next;
    else
        preth->next = tr->next;

    OslobodiVeze(tr->lista_clanova);
    free(tr->naziv);
    free(tr);
}


void Ispis(Takmicenje *takmicenja, Clan *clanovi)
{
    int max = -1;
```

```
    printf("TAKMICENJA KOJA NISU ZAVRSENA:\n");

    if (takmicenja == NULL)
        printf("Nema nezavrsenih takmicenja.\n");

    for (Takmicenje *t = takmicenja; t != NULL; t = t->next) {

        printf("ID: %d | %s | godina: %d | prijavljenih: %d | evidentirano: %d\n",
               t->id, t->naziv, t->godinaOdrzavanja,
               t->brojPrijavljenih, t->evidentirano);
    }

    for (Clan *c = clanovi; c != NULL; c = c->next) {
        if (c->br_takmicenja > max)
            max = c->br_takmicenja;
    }

    printf("\nCLAN SA NAJVECIM BROJEM TAKMICENJA:\n");

    for (Clan *c = clanovi; c != NULL; c = c->next) {
        if (c->br_takmicenja == max) {
            printf("ID: %d | %s | godina uclanjenja: %d | "
                   "ukupno poena: %d | broj takmicenja: %d\n",
                   c->id, c->ime_prezime,
                   c->godinaUclanjenja,
                   c->ukupnoPoena,
                   c->br_takmicenja);
        }
    }
}

int main()
{
    Clan *clanovi = UcitajClanove();

    Takmicenje *takmicenja = UcitajTakmicenja(clanovi);

    Azuriranje(clanovi, takmicenja);

    Takmicenje *trenutni = takmicenja;

    while (trenutni != NULL) {
        Takmicenje *sledeci = trenutni->next;

        if (trenutni->evidentirano ==
            trenutni->brojPrijavljenih) {

            ObrisiTakmicenje(&takmicenja, trenutni->id);
        }

        trenutni = sledeci;
    }

    Ispis(takmicenja, clanovi);

    OslobodiTakmicenja(takmicenja);
    OslobodiClanove(clanovi);

    return 0;
}
```