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

Jul, 2022

Poznat je lineup narednog IMI festivala. Za svakog izvođeča se zna njegov ID (ceo broj), ime (niz karaktera), bina (ceo broj od 1 do 5), kog dana festivala nastupa (ceo broj od 1 do 3), početak nastupa (sat i minut, dva cela broja) i vreme trajanja nastupa (broj minuta, ceo broj). Zna se da su svi ID-jevi različiti i da ne postoje dva izvođača koja istovremeno nastupaju na istoj bini. Podaci o izvođačima su već sortirani po danima, potom po binama, a potom po vremenu početka nastupa. Posetioci prave svoje planove za posetu festivalu. Za svakog posetioca se zna njegov ID (ceo broj), koliko izvođača želi da čuje (ceo broj) i za svakog izvođača koga želi da čuje njegov ID. Pretpostviti da će svako od posetilaca stići na festival 15 minuta pre početka nastupa izvođača koji je prvi kog želi da čuje tog dana i kada jednom uđe ostaje dok poslednji izvođač kog želi tog dana da čuje ne završi nastup. Karte se prodaju po danu, nezavisno. Učitati podatke o izvođačima i posetiocima, a potom odrediti ukupan broj prodatih karata i koliko su prosečno posetioci provodili na festivalu dnevno.
(3 poena main, 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 **UcitajIzvodjace** koja iz datoteke *Izvodjaci.txt* učitava podatke o izvođačima i formira listu/niz izvođača.
(3 poena, obavezno)
c) Napisati funkciju **NadjiIzvodjaca** koja za datu oznaku vraća pokazivač na traženog izvođača.
(2 poena)
d) Napisati funkciju **UcitajPosetioce** koja iz datoteke *Posetioci.txt* učitava podatke o posetiocima i formira listu/niz posetilaca.
(3 poena + 2 poena)
e) Napisati funkciju **Karte** koja za prosleđenog posetioca određuje koliko karata treba da kupi.
(3 poena)
f) Napisati funkciju **Vreme** koja za prosleđenog posetioca i redni broj dana festivala određuje koliko je tog dana posetilac proveo na festivalu.
(4 poena)
g) Napisati funkciju **KarteUkupno** koja određuje koliko karata je ukupno prodato.
(3 poena)
h) Napisati funkciju **VremeProsek** koja određuje koliko vremena su posetioci prosečno dnevno provodili vremena na festivalu.
(4 poena)

Pri učitavanju podataka podrazumevati da su svi podaci korektno zadati.
Dozvoljeno je proširivanje struktura i definisanje novih.
Maksimalan broj poena se dobija samo u slučaju da postoji veza između struktura koje čuvaju tačke i struktura koje čuvaju fugure, tj. ako postoje pokazivači “na jednu” ili “na obe strane”.

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 učutavanje stringova)**.**

Rešenje studenta 1

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

typedef struct izv{
	int id;
	char *ime;
	int bina;
	int dan;
	int sat;
	int minut;
	int trajanje;
	struct izv* sledeci;
}Izvodjac;

typedef struct el{
	Izvodjac* izvodjac;
	struct el* sledeci;
}Element;

typedef struct pos{
	int id;
	int broj_izvodjaca;
	Element* izvodjaci;
	int* vreme;
	int karte;
	struct pos* sledeci;
}Posetilac;

Izvodjac* UcitajIzvodjace(int *n)
{
	int i;
	FILE *f = fopen("Izvodjaci.txt", "r");
	Izvodjac *glava=NULL, *pom=NULL, *novi;

	fscanf(f, "%d", n);
	for(i=0; i<*n; i++)
	{
		novi = (Izvodjac*) malloc (sizeof(Izvodjac));
		fscanf(f, "%d", &(novi->id));
		fgetc(f);

		char unos[50];
		fgets(unos, 50, f);
		unos[strlen(unos)-1]='\0';
		novi->ime = (char*) malloc(sizeof(char)*strlen(unos));
		strcpy(novi->ime, unos);
		fscanf(f, "%d%d%d%d%d", &(novi->bina), &(novi->dan), &(novi->sat), &(novi->minut), &(novi-
>trajanje));

		novi->sledeci = NULL;

		if (glava==NULL)
		{
			glava=novi;
			pom=novi;
		}
		else
		{
			pom->sledeci=novi;
			pom=pom->sledeci;
		}
	}
	fclose(f);
	return glava;
}
```

```
Izvodjac* NadjiIzvodjaca(int id, Izvodjac* izvodjaci)
{
	while(izvodjaci && izvodjaci->id!=id)
	{
		izvodjaci = izvodjaci->sledeci;
	}
	return izvodjaci;
}

Posetilac* UcitajPosetioce(int *k, Izvodjac* glava_izvodjaca)
{
	int i,j;
	FILE *f = fopen("Posetioci.txt", "r");
	Posetilac* glava= NULL, *pom=NULL, *novi;
	fscanf(f,"%d", k);
	for(i=0; i<*k; i++)
	{
		Element *glavaE = NULL, *pomE=NULL, *noviE;

		novi= (Posetilac*) malloc (sizeof(Posetilac));
		fscanf(f, "%d%d", &(novi->id), &(novi->broj_izvodjaca));
		for(j=0; j<(novi->broj_izvodjaca); j++)
		{
			int broj;
			fscanf(f, "%d", &broj);
			noviE = (Element*) malloc(sizeof(Element));
			noviE ->izvodjac = NadjiIzvodjaca(broj, glava_izvodjaca);
			noviE ->sledeci = NULL;

			if (glavaE==NULL)
			{
				glavaE = noviE;
				pomE = noviE;
			}
			else
			{
				pomE ->sledeci = noviE;
				pomE = pomE ->sledeci;
			}
		}

		novi->izvodjaci = glavaE;
		novi->vreme = (int*) calloc(3,sizeof(int));
		novi->karte=0;
		novi->sledeci = NULL;

		if (glava==NULL)
		{
			glava = novi;
			pom = novi;
		}
		else
		{
			pom->sledeci = novi;
			pom = pom->sledeci;
		}
	}
	fclose(f);
	return glava;
}

void Karte(Posetilac *p)
{
	int i;
	Element *pom = p->izvodjaci;

	int *dani=(int*) calloc(3, sizeof(int));
	while(pom)
	{
		dani[(pom->izvodjac->dan)-1]+=1;
		pom = pom -> sledeci;
	}
```

```
	for(i=0; i<3; i++)
		if(dani[i]!=0)
			p->karte +=1;
}

void Vreme(Posetilac* p, int broj_dana)
{
	Element* pom = p->izvodjaci;
	int prvi_sat=23, prvi_minut=59, krajnji_sat=0, krajnji_minut=0;

	while(pom)
	{
		int sat = pom->izvodjac->sat;
		int minut = pom->izvodjac->minut;
		int trajanje = pom->izvodjac->trajanje;

		if(pom->izvodjac->dan == broj_dana)
		{
			if(sat<prvi_sat || (sat==prvi_sat && minut<prvi_minut))
			{
				prvi_sat = sat;
				prvi_minut = minut;
			}
			if( (sat+trajanje/60) > krajnji_sat || ( (sat+trajanje/60) == krajnji_sat &&
(minut+trajanje%60) > krajnji_minut) )
			{
				krajnji_sat = sat + trajanje/60;
				krajnji_minut = minut + trajanje%60;
			}
		}

		pom = pom->sledeci;
	}

	p->vreme[broj_dana-1]= abs(krajnji_sat - prvi_sat)*60 + abs(krajnji_minut-prvi_minut) + 15;
	if (p->vreme[broj_dana-1] == (23*60 + 59 + 15)) p->vreme[broj_dana-1]=0;
}

int KarteUkupno(Posetilac *glava_posetioca)
{
	int uk=0;
	while(glava_posetioca)
	{
		uk += glava_posetioca->karte;
		glava_posetioca = glava_posetioca -> sledeci;
	}
	return uk;
}

float VremeProsek(Posetilac *glava)
{
	int i;
	int suma = 0;
	int broj_dana = 0;

	while(glava)
	{
		for(i=0; i<3; i++)
			if (glava->vreme[i] != 0)
			{
				suma += glava->vreme[i];
				broj_dana++;
			}
		glava =  glava->sledeci;
	}
	return (1.0*suma)/broj_dana;
}

int main()
{
	int n,k,i;
	Izvodjac *izvodjaci;
```

```
	izvodjaci = UcitajIzvodjace(&n);
	Posetilac *posetioci, *pomP;
	pomP= (posetioci = UcitajPosetioce(&k, izvodjaci));


	while(pomP)
	{
		Karte(pomP);
		for(i=1; i<=3; i++) Vreme(pomP, i);
		pomP= pomP->sledeci;
	}

	printf("Prodato je ukupno %d karte\n", KarteUkupno(posetioci));
	printf("Prosecno vreme koje je posetilac dnevno provodio = %.2f u minutima.",
VremeProsek(posetioci));
}
```

Rešenje studenta 2

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

#define MAXLINE 50
#define TRAJANJE 3

struct Izvodjac{
	int id_izvodjaca;
	char *ime;
	int broj_bine;
	int dan_nastupa;
	int pocetak_nastupa_sati;
	int pocetak_nastupa_minuti;
	int vreme_trajanja;
	struct Izvodjac *sledeciIzvodjac;
};
struct Posetilac{
	int id_posetioca;
	int broj_izvodjaca;
	int broj_kupljenih_karata;
	int vreme[TRAJANJE];
	struct Izvodjac **zeljeniIzvodjaci;
	struct Posetilac *sledeciPosetilac;
};

struct Izvodjac *UcitajIzvodjace()
{
	struct Izvodjac *head = NULL;
	struct Izvodjac *noviIzvodjac, *pomIzvodjac;
	int brojIzvodjaca, i;
	char linija[MAXLINE];
	FILE *in;

	in = fopen("Izvodjaci.txt","r");
	if(in == NULL)
	{
		printf("Doslo je do greske pri otvaranju datoteke\n");
		exit(0);
	}
	fscanf(in,"%d",&brojIzvodjaca);
	for(i=0;i<brojIzvodjaca;i++)
	{
		noviIzvodjac = (struct Izvodjac*)malloc(sizeof(struct Izvodjac));
		if(noviIzvodjac == NULL)
		{
			printf("Doslo je do greske pri alokaciji memorije\n");
```

```
                        exit(0);
                }
                noviIzvodjac->sledeciIzvodjac = NULL;
                fscanf(in,"%d",&noviIzvodjac->id_izvodjaca);
                fgetc(in);
                fgets(linija,MAXLINE,in);
                if(linija[strlen(linija)-1] == '\n')
                        linija[strlen(linija)-1] = '\0';
                noviIzvodjac->ime = (char*)malloc(strlen(linija)*sizeof(char));
                if(noviIzvodjac->ime == NULL)
                {
                        printf("Doslo je do greske pri alokaciji memorije\n");
                        exit(0);
                }
                strcpy(noviIzvodjac->ime,linija);
                fscanf(in,"%d",&noviIzvodjac->broj_bine);
                fscanf(in,"%d",&noviIzvodjac->dan_nastupa);
                fscanf(in,"%d",&noviIzvodjac->pocetak_nastupa_sati);
                fscanf(in,"%d",&noviIzvodjac->pocetak_nastupa_minuti);
                fscanf(in,"%d",&noviIzvodjac->vreme_trajanja);

                if(head == NULL)
                {
                        head = noviIzvodjac;
                }
                else
                {
                        pomIzvodjac = head;
                        while(pomIzvodjac->sledeciIzvodjac != NULL)
                                pomIzvodjac = pomIzvodjac->sledeciIzvodjac;
                        pomIzvodjac->sledeciIzvodjac = noviIzvodjac;
                }
        }
        fclose(in);
        return head;
}
struct Izvodjac *NadjiIzvodjaca(struct Izvodjac *head, int oznaka)
{
        struct Izvodjac *pom;

        pom = head;
        while(pom != NULL)
        {
                if(pom->id_izvodjaca == oznaka)
                        return pom;
                pom = pom->sledeciIzvodjac;
        }
        return pom;
}
int Karte(struct Posetilac *posetilac)
{
        int *niz;
        int i, indeks, broj_karata = 0;

        niz = (int*)calloc(TRAJANJE,sizeof(int));
        if(niz == NULL)
        {
                printf("Doslo je do greske pri alokaciji memorije\n");
                exit(0);
        }
        for(i=0;i<posetilac->broj_izvodjaca;i++)
        {
                indeks = posetilac->zeljeniIzvodjaci[i]->dan_nastupa;
                if(niz[indeks-1] == 0)
                        niz[indeks-1] = 1;
        }
        for(i=0;i<TRAJANJE;i++)
        {
                if(niz[i] == 1)
                        broj_karata++;
        }
        return broj_karata;
}
```

```
int Vreme(struct Posetilac *posetilac, int dan)
{
	struct Izvodjac *prvi = NULL;
	struct Izvodjac *poslednji = NULL;
	int i,vreme1,vreme2;

	for(i=0;i<posetilac->broj_izvodjaca;i++)
	{
		if(posetilac->zeljeniIzvodjaci[i]->dan_nastupa == dan)
		{
			if(prvi == NULL)
				prvi = posetilac->zeljeniIzvodjaci[i];
			else
			{
				vreme1 = prvi->pocetak_nastupa_sati*60+prvi->pocetak_nastupa_minuti;
				vreme2 = posetilac->zeljeniIzvodjaci[i]->pocetak_nastupa_sati*60+posetilac-
>zeljeniIzvodjaci[i]->pocetak_nastupa_minuti;
				if(vreme2 < vreme1)
					prvi = posetilac->zeljeniIzvodjaci[i];
			}
		}
	}
	if(prvi == NULL)
		return 0;
	for(i=0;i<posetilac->broj_izvodjaca;i++)
	{
		if(posetilac->zeljeniIzvodjaci[i]->dan_nastupa == dan)
		{
			if(poslednji == NULL)
				poslednji = posetilac->zeljeniIzvodjaci[i];
			else
			{
				vreme1 = poslednji->pocetak_nastupa_sati*60+poslednji-
>pocetak_nastupa_minuti+poslednji->vreme_trajanja;
				vreme2 = posetilac->zeljeniIzvodjaci[i]->pocetak_nastupa_sati*60+posetilac-
>zeljeniIzvodjaci[i]->pocetak_nastupa_minuti+posetilac->zeljeniIzvodjaci[i]->vreme_trajanja;
				if(vreme2 > vreme1)
					poslednji = posetilac->zeljeniIzvodjaci[i];

			}
		}
	}
	if(prvi == poslednji)
	{
		return prvi->vreme_trajanja+15;
	}
	vreme1 = poslednji->pocetak_nastupa_sati*60+poslednji->pocetak_nastupa_minuti+poslednji-
>vreme_trajanja - prvi->pocetak_nastupa_sati*60 - prvi->pocetak_nastupa_minuti + 15;
	return vreme1;
}
struct Posetilac *UcitajPosetioce(struct Izvodjac *glava)
{
	struct Posetilac *head = NULL;
	struct Posetilac *noviPosetilac, *pomPosetilac;
	struct Izvodjac *pronadjen;
	int brojPosetilaca,i,j;
	int id_izvodjac, brojKarata, vremeDan;
	FILE *in;

	in = fopen("Posetioci.txt","r");
	if(in == NULL)
	{
		printf("Doslo je do greske pri otvaranju datoteke\n");
		exit(0);
	}
	fscanf(in,"%d",&brojPosetilaca);
	for(i=0;i<brojPosetilaca;i++)
	{
		noviPosetilac = (struct Posetilac*)malloc(sizeof(struct Posetilac));
		if(noviPosetilac == NULL)
		{
			printf("Doslo je do greske pri alokaciji memorije\n");
			exit(0);
```

```
		}
		noviPosetilac->sledeciPosetilac = NULL;
		fscanf(in,"%d",&noviPosetilac->id_posetioca);
		fscanf(in,"%d",&noviPosetilac->broj_izvodjaca);
		noviPosetilac->zeljeniIzvodjaci = (struct Izvodjac**)malloc(noviPosetilac-
>broj_izvodjaca*sizeof(struct Izvodjac*));
		if(noviPosetilac->zeljeniIzvodjaci == NULL)
		{
			printf("Doslo je do greske pri alokaciji memorije\n");
			exit(0);
		}
		for(j=0;j<noviPosetilac->broj_izvodjaca;j++)
		{
			fscanf(in,"%d",&id_izvodjac);
			pronadjen = NadjiIzvodjaca(glava,id_izvodjac);
			noviPosetilac->zeljeniIzvodjaci[j] = pronadjen;
		}
		brojKarata = Karte(noviPosetilac);
		noviPosetilac->broj_kupljenih_karata = brojKarata;

		for(j=0;j<TRAJANJE;j++)
		{
			vremeDan = Vreme(noviPosetilac,j+1);
			noviPosetilac->vreme[j] = vremeDan;
		}

		if(head == NULL)
			head = noviPosetilac;
		else
		{
			pomPosetilac = head;
			while(pomPosetilac->sledeciPosetilac != NULL)
				pomPosetilac = pomPosetilac->sledeciPosetilac;
			pomPosetilac->sledeciPosetilac = noviPosetilac;
		}
	}
	fclose(in);
	return head;
}
int KarteUkupno(struct Posetilac *head)
{
	struct Posetilac *pom;
	int ukupno = 0;

	pom = head;
	while(pom != NULL)
	{
		ukupno += pom->broj_kupljenih_karata;
		pom = pom->sledeciPosetilac;
	}
	return ukupno;
}
float VremeProsek(struct Posetilac *head)
{
	int i, brojac, ukupno = 0;
	struct Posetilac *pom;
	float prosek;

	brojac = 0;
	for(i=0;i<TRAJANJE;i++)
	{
		pom = head;
		while(pom != NULL)
		{
			if(pom->vreme[i] != 0)
			{
				brojac++;
				ukupno += pom->vreme[i];
			}
			pom = pom->sledeciPosetilac;
		}
	}
	if(brojac == 0)
```

```
                return 0.0;
        prosek = (float)ukupno/brojac;
        return prosek;
}
int main()
{
        struct Izvodjac *izvodjaci;
        struct Posetilac *posetioci;
        int ukupnoKarata;
        float prosecnoVreme;

        izvodjaci = UcitajIzvodjace();
        posetioci = UcitajPosetioce(izvodjaci);
        ukupnoKarata = KarteUkupno(posetioci);
        printf("Ukupno prodatih karata: %d\n",ukupnoKarata);
        prosecnoVreme = VremeProsek(posetioci);
        printf("Prosecno vreme: %.2f minuta\n",prosecnoVreme);
}
```