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

Jul, 2024

U banci svaki od šaltera ima svoju uslugu koju pruža, pri čemu više šaltera može pružati istu uslugu. Za svaki šalter se unosi ID šaltera (ceo broj), naziv usluge koju pruža (niz karaktera) i očekivano vreme po klijentu za pružanje navedene usluge (ceo broj, vreme u minutima). Šalteri koju pružaju istu uslugu ne moraju imati isto očekivano vreme. Radno vreme svih šaltera je od 8-20h. Klijenti prilikom dolaska u banku se izjašnjavaju koje usluge su im potrebne. Jedan klijent može imati zahtev za više usluga, pri čemu se svaka pruža na različitom šalteru i neće postojati dva zahteva za istu uslugu. Klijenti se obrađuju po vremenu pristizanja i za svakog klijenta se unosi njegov ID (ceo broj), vreme dolaska (dva cela broja – sat i minut), koliko usluga mu je potrebno i za svaku uslugu naziv usluge. Klijent će biti raspoređen na šalter za odgovarajuću uslugu koji će prvi biti slobodan. Klijent koji ne može za neku od prijavljenih usluga biti uslužen do kraja radnog vremena, za tu uslugu će biti odbijen. Za uneti ID šaltera ispisati sve ID-jeve klijenata kojima je usluga pružena na tom šalteru i ukupno vreme bez klijenata na šalteru. Ispisati podatke klijenta koji je proveo najviše vremena u banci od dolaska do završetka svih usluga.
(kompletna main funkcija - 3 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 **UcitajSaltere** koja iz datoteke *Salteri.txt* učitava podatke o šalterima i formira listu/niz šaltera.
(3 poena, obavezno)
c) Napisati funkciju **NadjiSalter** koja za datu namenu vraća pokazivač na šalter te namene koji će prvi biti slobodan.
(4 poena)
d) Napisati funkciju **UcitajKlijente** koja iz datoteke *Klijenti.txt* učitava podatke o klijentima, formira listu/niz klijenata (ili više lista/nizova) i vrši raspoređivanje po šalterima.
(4 + 4 poena)
Napomena. Rapoređivanje se može izdvojiti u posebnu funkciju**.** Biće priznato i ako se klijentu sa više prijavljenih usluga pruža jedna po jedna usluga i ako ima preklapanja, pa je istovremeno na „više šaltera".
e) Napisati funkciju **Ispis** koja za dati ID šaltera ispisuje sve klijente koji su bili usluženi na tom šalteru i ukupno vreme na šalteru bez klijenata.
(2+3 poena)
f) Napisati funkciju **Klijent** koja određuje klijenta koji je proveo najviše vremena u banci, od prijave do završetka usluga.
(4 poena)

Pri učitavanju podataka podrazumevati da su svi podaci korektno zadati.
Podaci iz datoteke se mogu učitavati uz pretpostvku 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 šaltere i struktura koje čuvaju klijente, tj. ako postoje pokazivači "na jednu" ili "na obe strane".
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 učutavanje 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>

typedef struct Salter {
	int ID;
	char *naziv_usluge;
	int vreme_po_klijentu;
	int vreme_bez_klijenata;
	int zauzet_do;
	struct Salter *sledeci;
}Salter;

typedef struct Usluga {
	char *naziv_usluge;
	Salter *salter;
	struct Usluga *sledeci;
}Usluga;

typedef struct Klijent {
	int ID;
	int dolazak;
	int vreme_zauzet;
	int brUsluga;
	Usluga *usluge;
	struct Klijent *sledeci;
}Klijent;

Salter *UcitajSaltere()
{
	FILE *f = fopen("salteri.txt", "r");
	Salter *glava = NULL;
	while(!feof(f))
	{
		Salter *novi = (Salter *) malloc(sizeof(Salter));
		fscanf(f, "%d", &novi->ID);
		char buffer[255];
		fgetc(f);
		fgets(buffer, 255, f);
		buffer[strlen(buffer) - 1] = '\0';
		novi->naziv_usluge = (char *) malloc(strlen(buffer) + 1);
		strcpy(novi->naziv_usluge, buffer);
		fscanf(f, "%d", &novi->vreme_po_klijentu);
		novi->vreme_bez_klijenata = 720; //720 minuta = 12 sati
		novi->zauzet_do = 0; //ako je salter slobodan 0, inace do kad je zauzet
		novi->sledeci = glava;
		glava = novi;
	}
	fclose(f);
	return glava;
}

Salter *NadjiSalter(Salter *glava, char *namena, int vreme)
{
	Salter *pom = NULL;
	while (glava != NULL)
	{
		if (strcmp(glava->naziv_usluge, namena) != 0) glava = glava->sledeci;
		else if (glava->zauzet_do > vreme) //ako je salter za tu namenu ali zauzet
		{
			//da li ce salter biti slobodan za manje vreme?
			if (pom != NULL && ((glava->zauzet_do - vreme) < (pom->zauzet_do - vreme)))
pom = glava;
```

```
                else pom = glava;
            }
            else return glava; //ako ima slobodan salter vrati slobodan
        }
        return pom; //ako nema slobodan vraca prvi salter koji ce biti slobodan
}

Klijent *UcitajKlijente(Salter *glavaS)
{
        FILE *f = fopen("klijenti.txt", "r");
        Klijent *glavaK = NULL;
        while(!feof(f))
        {
            Klijent *novi = (Klijent *) malloc(sizeof(Klijent));
            int vreme_sat, vreme_min;
            fscanf(f, "%d%d%d%d", &novi->ID, &vreme_sat, &vreme_min, &novi->brUsluga);
            vreme_sat = vreme_sat * 60 + vreme_min;
            novi->dolazak = vreme_sat;
            novi->vreme_zauzet = 0; //vreme provedeno sa klijentima
            novi->usluge = NULL;
            int i;
            for (i = 0; i < novi->brUsluga; i++)
            {
                Usluga *nova = (Usluga *) malloc(sizeof(Usluga));
                char buffer[255];
                fgetc(f);
                fgets(buffer, 255, f);
                buffer[strlen(buffer) - 1] = '\0';
                nova->naziv_usluge = (char *) malloc(strlen(buffer) + 1);
                strcpy(nova->naziv_usluge, buffer);
                nova->salter = NadjiSalter(glavaS, nova->naziv_usluge, novi->dolazak);
                if (nova->salter != NULL) //postoji salter sa uslugom
                {
                    //ako usluga traje posle 8 uvece (1200 minuta) klijent ce biti
odbijen
                    if ((novi->dolazak + nova->salter->vreme_po_klijentu) > 1200) nova-
>salter = NULL;
                    else
                    {
                        if ((nova->salter)->zauzet_do > novi->dolazak)
                        {
                            novi->dolazak = nova->salter->zauzet_do + nova->salter-
>vreme_po_klijentu;
                            novi->vreme_zauzet += nova->salter->vreme_po_klijentu;
                        }
                        else
                        {
                            novi->dolazak += nova->salter->vreme_po_klijentu;
                            novi->vreme_zauzet += nova->salter->vreme_po_klijentu;
                        }
                        nova->salter->zauzet_do = novi->dolazak;
                        nova->salter->vreme_bez_klijenata -= nova->salter-
>vreme_po_klijentu;
                    }
                }
                nova->sledeci = novi->usluge;
                novi->usluge = nova;
            }
            novi->sledeci = glavaK;
            glavaK = novi;
        }
        fclose(f);
        return glavaK;
}
```

```
void KlijentInfo(Klijent *glava)
{
	if (glava == NULL) printf("Nema klijenata\n");
	else
	{
		Klijent *pom = glava;
		while (glava != NULL) {
			if (glava->vreme_zauzet > pom->vreme_zauzet) pom = glava;
			glava = glava->sledeci;
		}
		printf("Najvise vremena na salterima je proveo klijent %d", pom->ID);
	}
}

void Ispis(int ID, Salter *glavaS, Klijent *glavaK)
{
	while(glavaS != NULL && glavaS->ID != ID) glavaS = glavaS->sledeci;
	if (glavaS == NULL) {
		printf("Salter sa ID %d ne postoji\n", ID);
		return;
	}
	printf("Salter %d: %d minuta bez klijenata\n", glavaS->ID, glavaS->vreme_bez_klijenata);

	Klijent *klijent = glavaK;
    while (klijent != NULL)
	{
	  Usluga *usluga = klijent->usluge;
	  while (usluga != NULL)
		{
		if (usluga->salter != NULL && usluga->salter->ID == ID)
			{
		    printf("%d - klijent %d\n", usluga->salter->ID, klijent->ID);
		    break;
		}
		usluga = usluga->sledeci;
	  }
	  klijent = klijent->sledeci;
    }
}

int main()
{
	Salter *glavaS = UcitajSaltere();
	Klijent *glavaK = UcitajKlijente(glavaS);
	Salter *pom = glavaS;
	while(pom != NULL)
	{
		Ispis(pom->ID, glavaS, glavaK);
		pom = pom->sledeci;
	}
	KlijentInfo(glavaK);

	return 0;
}
```