



## ПРОГРАМИРАЊЕ I

### Практични део испита

Име и презиме: \_

Бр. индекса: \_

Поени: \_

#### Обавештења:

- Испит траје 120 минута.
- На локалном диску С направити фолдер под називом JANUAR\_P1T1.
- НАЗИВ ФАЈЛА **ИЗВОРНОГ КОДА** МОРА БИТИ У СЛЕДЕЋЕМ ФОРМАТУ име\_презиме\_годинауписа\_бројиндекса (пример: Pera\_Peric\_2021\_0001.c). Сви задаци се раде у једном пројекту, у једној .c датотеци.
- За сваки задатак где је дат прототип функције мора да се имплементира функција која одговара прототипу (ИСТИ назив, листа параметара и повратна вредност), исто важи и за дате типове.
- Могу се дописивати додатне помоћне функције и типови.

#### Задатак:

Потребно је имплементирати следеће функције:

Дата је текстуална датотека која садржи податке о дневној продатој количини производа за сваки дан у недељи. Подаци се налазе у следећем формату:

*шифра производа кол\_пон кол\_уто кол\_сре кол\_чет кол\_пет*

Пример такве датотеке:

1113 5 10 12 3 1

2226 10 2 4 5 8

3339 9 20 3 1 1

Имплементирати следеће функције:

▪ **void procitaj\_datoteku(char \* naziv, int info[][6], int \* n)** 15п.

Функција чита податке из текстуалне датотеке и уписује их матрицу. Садржај матрице треба да одговара садржају датотеке, тако да се у првој колони налазе шифре производа, у другој продата количина за понедељак итд.

Матрица увек има тачно шест колона (прва је за шифру, остале за сваки радни дан у недељи).

**Број редова није унапред познат (број редова представља n)**

Садржај матрице након извршавања функције (за дати пример текстуалне датотеке):

	0	1	2	3	4	5
0	1113	5	10	12	3	1
1	2226	10	2	4	5	8
2	3339	9	20	3	1	1

Број редова матрице (n) треба ажурирати да буде три (за овај пример).

▪ **int ukupna\_kolicina(int index\_reda, int info[][6], int n)** 10п.

Функција враћа укупну продату количину за производ који се налази у реду са прослеђеним бројем индекса.

За *index\_reda* 1 и горе прослеђену матрицу, функција треба да врати 29.

▪ **int azuriraj\_kolicinu(int nova\_kol, int sifra\_pr, int dan, int info[][6], int n)** 10п.

Функција треба да омогући ажурирање количине за дати производ и за дати дан.

За *nova\_kol* = 5, *sifra\_pr* = 3339, *dan* = 2 и прослеђену матрицу, матрицу треба ажурирати на следећи начин:

	0	1	2	3	4	5
0	1113	5	10	12	3	1
1	2226	10	2	4	5	8
2	3339	9	5	3	1	1

Што значи да су вредности за дане следећи:

1 – понедељак, 2 – уторак, 3 – среда, 4 – четвртак, 5 – петак.



Дати су следећи типови:

```
typedef struct cvor CVOR;
typedef struct cvor * PCVOR;
struct cvor{
    PRODAJA prodaja;
    PCVOR sledeci;
};
typedef struct prodaja {
    int sifra_proizvoda;
    int prodata_kolicina;
    char dan[10];
} PRODAJA;
```

- **void dodaj(PCVOR \* glava, int sifra\_proizvoda, int prodata\_kolicina, char dan[])** 10п.

Функција додаје нови чвор на крај листе.

(Уколико се имплементира додавање на почетак листе бодује се са 5п.)

- **void kreiraj\_listu(int info[][6], char dan[], PCVOR \* glava, int n)** 15п.

Функција пребацује податке о продаји сваког производа за прослеђени дан из матрице у листу (искористити функцију *dodaj()*). Сваки ред у матрици треба превести у један чвор тако да шифра производа одговара првој вредности у реду матрице, за продату количину треба узети количину која је продата тог дана, док се за дан уписује скраћен назив дана ("pon", "uto", "sre", "cet", "pet").

Могуће вредности за дан су "pon", "uto", "sre", "cet", "pet".

Уколико је прослеђена вредност за дан = "sre", у листи треба да се налазе следећи чланови:

1113	→	2226	→	3336
12		4		3
"sre"		"sre"		"sre"

- **void sortiraj(PCVOR glava)** 15п.

Функција сортира листу по количини продатих производа у растућем редоследу.

- **PRODAJA pronadji(int sifra, PCVOR glava)** 10п.

Функција проналази и враћа производ са прослеђеном шифром.

Уколико производ са том шифром не постоји враћа производ са шифром -999 и продатом количином -999.

- **int prover\_i\_sifru(int sifra)** 15п.

Функција проверава да ли је шифра у добром формату, уколико јесте враћа 1, а уколико није онда треба да врати 0. Шифра је у добром формату уколико последња (контролна) цифра одговара збиру свих претходних цифара. Уколико је збир цифара двоцифрени број, потребно је сабрати цифре тог збира и онда тај број представља контролну цифру.

Например:

1113 је у добром формату јер  $1+1+1=3$

8212 је у добром формату јер  $8+2+1=11$   $1+1=2$

```

#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include <math.h>

//JANUAR 2023 - GRUPA 1

typedef struct prodaja {
    int sifra_proizvoda;
    int prodata_kolicina;
    char dan[10];
} PRODAJA;

typedef struct cvor {
    PRODAJA prodaja;
    struct cvor* sledeci;
} CVOR;

typedef struct cvor* PCVOR;

//GLAVNE FUNKCIJE

void procitaj_datoteku(char* naziv, int info[][6], int* n) {
    FILE* datoteka = fopen(naziv, "r");
    if (datoteka == NULL) {
        printf("Greska pri otvaranju datoteke!\n");
        return;
    }

    int i = 0;
    while (!feof(datoteka)) {
        fscanf(datoteka, "%d %d %d %d %d %d", &info[i][0], &info[i][1],
&info[i][2], &info[i][3], &info[i][4], &info[i][5]);
        i++;
    }

    *n = i;

    fclose(datoteka);
}

int ukupna_kolicina(int index_reda, int info[][6], int n) {
    int suma = 0;

    for (int i = 1; i < 6; i++) {
        suma += info[index_reda][i];
    }

    return suma;
}

int azuriraj_kolicinu(int nova_kol, int sifra_pr, int dan, int info[][6],
int n) {
    for (int i = 0; i < n; i++) {
        if (info[i][0] == sifra_pr) {
            info[i][dan] = nova_kol;
            return 1;
        }
    }
}

```

```

    return 0;
}

void dodaj(PCVOR* glava, int sifra_proizvoda, int prodata_kolicina, char
dan[]) {
    PCVOR noviCvor = malloc(sizeof(CVOR));
    noviCvor->prodaja.sifra_proizvoda = sifra_proizvoda;
    noviCvor->prodaja.prodata_kolicina = prodata_kolicina;
    strcpy(noviCvor->prodaja.dan, dan);
    noviCvor->sledeci = NULL;

    if (*glava == NULL) {
        *glava = noviCvor;
    }
    else {
        PCVOR trenutni = *glava;
        while (trenutni->sledeci != NULL) {
            trenutni = trenutni->sledeci;
        }
        trenutni->sledeci = noviCvor;
    }
}

void kreiraj_listu(int info[][6], char dan[], PCVOR* glava, int n) {
    for (int i = 0; i < n; i++) {
        if (strcmp(dan, "pon") == 0) {
            dodaj(glava, info[i][0], info[i][1], dan);
        }
        if (strcmp(dan, "uto") == 0) {
            dodaj(glava, info[i][0], info[i][2], dan);
        }
        if (strcmp(dan, "sre") == 0) {
            dodaj(glava, info[i][0], info[i][3], dan);
        }
        if (strcmp(dan, "cet") == 0) {
            dodaj(glava, info[i][0], info[i][4], dan);
        }
        if (strcmp(dan, "pet") == 0) {
            dodaj(glava, info[i][0], info[i][5], dan);
        }
    }
}

void sortiraj(PCVOR glava) {
    int zamenjeno;
    PCVOR trenutni;
    PCVOR prethodni = NULL;

    if (glava == NULL) {
        return;
    }

    do {
        zamenjeno = 0;
        trenutni = glava;

        while (trenutni->sledeci != prethodni) {
            if (trenutni->prodaja.prodata_kolicina > trenutni->sledeci-
>prodaja.prodata_kolicina) {

```

```

        PRODAJA privremeni = trenutni->prodaja;
        trenutni->prodaja = trenutni->sledeci->prodaja;
        trenutni->sledeci->prodaja = privremeni;
        zamenjeno = 1;
    }

    trenutni = trenutni->sledeci;
}

prethodni = trenutni;
} while (zamenjeno);
}

PRODAJA pronadji(int sifra, PCVOR glava) {
    while (glava != NULL) {
        if (glava->prodaja.sifra_proizvoda == sifra) {
            return glava->prodaja;
        }
        glava = glava->sledeci;
    }

    PRODAJA nepostojeci;
    nepostojeci.sifra_proizvoda = -999;
    nepostojeci.prodata_kolicina = -999;
    return nepostojeci;
}

int prover_i_sifru(int sifra) {
    int prva = sifra / 1000;
    int druga = (sifra % 1000) / 100;
    int treca = (sifra % 100) / 10;
    int cetvrta = sifra % 10;
    if (prva + druga + treca < 10) {
        if (prva + druga + treca == cetvrta)
            return 1;
    }
    else if (prva + druga + treca >= 10) {
        int temp = prva + druga + treca;
        int prval = temp % 10;
        int drugal = temp / 10;
        if (prval + drugal == cetvrta) {
            return 1;
        }
    }
    return 0;
}

//POMOCNE FUNKCIJE

void prikaziMatricu(int info[][6], int n) {
    if (n == 0) {
        printf("Matrica je prazna.\n");
        return;
    }

    printf("Matrica:\n");
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < 6; j++) {
            printf("%d ", info[i][j]);

```

```

        }
        printf("\n");
    }
}

void prikaziListu(PCVOR glava) {
    if (glava == NULL) {
        printf("Lista je prazna.\n");
        return;
    }

    printf("\nLista:\n");
    PCVOR trenutni = glava;
    while (trenutni != NULL) {
        printf("%d %d %s\n", trenutni->prodaja.sifra_proizvoda, trenutni->prodaja.prodata_kolicina, trenutni->prodaja.dan);
        trenutni = trenutni->sledeci;
    }
}

int main() {
    int info[100][6];
    int n = 0;

    //1. Ucitavanje podataka u matricu
    procitaj_datoteku("podaci.txt", info, &n);
    prikaziMatricu(info, n);

    //2. Ukupna kolicina proizvoda
    printf("\nUkupna kolicina za index reda 1: %d\n\n",
    ukupna_kolicina(1, info, n));

    //3. Azurirana kolicina
    azuriraj_kolicinu(5, 3339, 2, info, n);
    prikaziMatricu(info, n);

    //4. Kreiranje liste
    PCVOR glava = NULL;
    dodaj(&glava, 1111, 15, "pon");
    dodaj(&glava, 1221, 18, "uto");
    dodaj(&glava, 1331, 12, "sre");
    dodaj(&glava, 1331, 36, "sre");
    dodaj(&glava, 1551, 27, "cet");
    kreiraj_listu(info, "sre", &glava, n);
    prikaziListu(glava);

    //5. Sortiranje liste
    sortiraj(glava);
    prikaziListu(glava);

    //6. Pronadji proizvod
    PRODAJA pronadjena = pronadji(3339, glava);
    if (pronadjena.sifra_proizvoda == -999) {
        printf("\nProizvod sa zadatom sifrom ne postoji.\n");
    }
    else {
        printf("\nPronadjen proizvod:\n");
        printf("%d %d %s\n", pronadjena.sifra_proizvoda,
        pronadjena.prodata_kolicina, pronadjena.dan);
    }
}

```

```
//7. Provera sifre
int sifra = 1113;
if (proveri_sifru(sifra)) {
    printf("\nSifra %d je u dobrom formatu.\n", sifra);
}
else {
    printf("\nSifra %d nije u dobrom formatu.\n", sifra);
}

return 0;
}
```



## ПРОГРАМИРАЊЕ I

### Практични део испита

Име и презиме: \_

Бр. индекса: \_

Поени: \_

#### Обавештења:

- Испит траје 120 минута.
- На локалном диску С направити фолдер под називом JANUAR\_P1T2.
- НАЗИВ ФАЈЛА **ИЗВОРНОГ КОДА** МОРА БИТИ У СЛЕДЕЋЕМ ФОРМАТУ име\_презиме\_годинауписа\_бројиндекса (пример: Pera\_Peric\_2021\_0001.c). Сви задаци се раде у једном пројекту, у једној .c датотеци.
- За сваки задатак где је дат прототип функције мора да се имплементира функција која одговара прототипу (ИСТИ назив, листа параметара и повратна вредност), исто важи и за дате типове.
- Могу се дописивати додатне помоћне функције и типови.

#### Задатак:

Потребно је имплементирати следеће функције:

Дата је текстуална датотека која садржи податке о положеним испитима студената, у следећем формату:  
*број\_индекса година\_уписа шифра\_предмета оцена*

Пример такве датотеке:

```
26 2018 1113 9
116 2019 2220 7
1052 2020 28310 10
113 2019 2220 8
2 2022 1113 10
```

Имплементирати следеће функције:

▪ **void procitaj\_datoteku(char \* naziv, int info[][4], int \* n)** 15п.

Функција чита податке из текстуалне датотеке и уписује их матрицу. Садржај матрице треба да одговара садржају датотеке, тако да се у првој колони налази број индекса, у другој година уписа, у трећој шифра предмета и у четвртој оцена.

Матрица увек има тачно четири колоне.

**Број редова није унапред познат (број редова представља n).**

Садржај матрице након извршавања функције (за дати пример текстуалне датотеке):

	0	1	2	3
0	26	2018	1113	9
1	116	2019	2220	7
2	1052	2020	28310	10
3	113	2019	2220	8
4	2	2022	1113	10

Број редова матрице треба ажурирати да буде 5 (за овај пример).

▪ **double prosecna\_ocena\_predmet(int sifra\_predmeta, int info[][4], int n)** 10п.

Функција враћа просечну оцену за предмет са прослеђеном шифром.

▪ **void prikazi(int sifra\_predmeta, int info[][4], int n)** 10п.

Функција приказује индексе и оцене свих студената који су положили предмет са прослеђеном шифром предмета у следећем формату (пример је дат за прослеђену шифру предмета 1113):

1. 2018/26 1113 9
2. 2022/2 1113 10

Дати су следећи типови:

```
typedef struct cvor CVOR;
typedef struct cvor * PCVOR;
struct cvor{
    PRIJAVA prijava;
    PCVOR sledeci;
}
```





```
};
typedef struct prijava{
    int broj_indeksa;
    int godina_upisa;
    int sifra_predmeta;
    int ocena;
}PRIJAVA;
▪ void dodaj(PCVOR * glava, int broj_indeksa, int godina_upisa, int ocean, int sifra_predmeta) 10п.
```

Функција додаје нови чвор на крај листе.  
(Уколико се имплементира додавање на почетак листе, функција се бодује са 5п.)

```
▪ int da_li_postoji(PCVOR glava, int broj_indeksa, int godina_upisa, int sifra_predmeta) 15п.
```

Функција проверава да ли постоји положен испит са прослеђеном шифром предмета за студента са прослеђеним индексом (бројем индекса и годином уписа). Уколико постоји функција враћа оцену коју је студент добио. Уколико не постоји, функција треба да врати -1.

```
▪ void kreiraj_listu(int info[][4], int godina_upisa, PCVOR * glava, int n) 10п.
```

Функција пребацује податке из прослеђене матрице у листу (искористити функцију *dodaj()*), тако што у листу убацује све положене испите студената чија је година уписа једнака прослеђеној.  
Приликом креирања листе потребно је проверити да не дође до понављања, један студент (број индекса и година уписа) може само једном да положи један испит (шифра предмета).  
(Уколико се приликом креирања листе не проверава услов који се односи на дупликате, онда се бодује са 5п.)

```
▪ void sortiraj(PCVOR glava) 15п.
```

Функција сортира листу по оцени у опадајућем редоследу.

```
▪ int prover_i_sifru(int sifra_predmeta) 15п.
```

Функција проверава да ли је шифра у добром формату, уколико јесте функција треба да врати 1, уколико није онда треба да врати 0. Шифра је у добром формату уколико је последња (контролна) цифра једнака 1 ако је збир свих претходних цифара непаран, односно 0 уколико је збир паран.

Например:  
1111 : 1+1+1=3 -> 3 није паран број -> последња цифра је 1 -> шифра је у добром формату  
28310 : 2+8+3+1=14 -> 14 је паран број -> последња цифра је 0 -> шифра је у добром формату

```

#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include <math.h>

//JANUAR 2023 - GRUPA 2

typedef struct prijava {
    int broj_indeksa;
    int godina_upisa;
    int sifra_predmeta;
    int ocena;
} PRIJAVA;

typedef struct cvor {
    PRIJAVA prijava;
    struct cvor* sledeci;
} CVOR;

typedef struct cvor* PCVOR;

//GLAVNE FUNKCIJE

void procitaj_datoteku(char* naziv, int info[][4], int* n) {
    FILE* datoteka = fopen(naziv, "r");
    if (datoteka == NULL) {
        printf("Greska pri otvaranju datoteke.\n");
        return;
    }

    int i = 0;
    while (fscanf(datoteka, "%d %d %d %d", &info[i][0], &info[i][1],
&info[i][2], &info[i][3]) == 4) {
        i++;
    }

    *n = i;

    fclose(datoteka);
}

double prosecna_ocena_predmet(int sifra_predmeta, int info[][4], int n) {
    int suma_ocena = 0;
    int broj_studenata = 0;

    for (int i = 0; i < n; i++) {
        if (info[i][2] == sifra_predmeta) {
            suma_ocena += info[i][3];
            broj_studenata++;
        }
    }

    if (broj_studenata > 0) {
        return (double)suma_ocena / broj_studenata;
    }
    else {
        return 0.0;
    }
}

```

```

void prikazi(int sifra_predmeta, int info[][4], int n) {
    printf("Indeksi i ocene studenata za predmet sa sifrom %d:\n",
sifra_predmeta);

    for (int i = 0; i < n; i++) {
        if (info[i][2] == sifra_predmeta) {
            printf("%d. %d/%d %d %d\n", i + 1, info[i][1], info[i][0],
info[i][2], info[i][3]);
        }
    }
}

void dodaj(PCVOR* glava, int broj_indeksa, int godina_upisa, int ocena,
int sifra_predmeta) {
    PCVOR noviCvor = malloc(sizeof(CVOR));
    noviCvor->prijava.broj_indeksa = broj_indeksa;
    noviCvor->prijava.godina_upisa = godina_upisa;
    noviCvor->prijava.sifra_predmeta = sifra_predmeta;
    noviCvor->prijava.ocena = ocena;
    noviCvor->sledeci = NULL;

    if (*glava == NULL) {
        *glava = noviCvor;
    }
    else {
        PCVOR trenutni = *glava;
        while (trenutni->sledeci != NULL) {
            trenutni = trenutni->sledeci;
        }
        trenutni->sledeci = noviCvor;
    }
}

int da_li_postoji(PCVOR glava, int broj_indeksa, int godina_upisa, int
sifra_predmeta) {
    PCVOR trenutni = glava;
    while (trenutni != NULL) {
        if (trenutni->prijava.broj_indeksa == broj_indeksa &&
            trenutni->prijava.godina_upisa == godina_upisa &&
            trenutni->prijava.sifra_predmeta == sifra_predmeta) {
            return trenutni->prijava.ocena;
        }
        trenutni = trenutni->sledeci;
    }
    return -1;
}

void kreiraj_listu(int info[][4], int godina_upisa, PCVOR* glava, int n)
{
    for (int i = 0; i < n; i++) {
        if (info[i][1] == godina_upisa) {
            int broj_indeksa = info[i][0];
            int godina_upisa = info[i][1];
            int sifra_predmeta = info[i][2];
            int ocena = info[i][3];

            if (da_li_postoji(*glava, broj_indeksa, godina_upisa,
sifra_predmeta) == -1) {

```

```

        dodaj(glava, broj_indeksa, godina_upisa, ocena,
sifra_predmeta);
    }
}

```

```

void sortiraj(PCVOR glava) {
    int promena;
    PCVOR trenutni;
    PCVOR prethodni = NULL;

    if (glava == NULL) {
        return;
    }

    do {
        promena = 0;
        trenutni = glava;

        while (trenutni->sledeci != prethodni) {
            if (trenutni->prijava.ocena < trenutni->sledeci-
>prijava.ocena) {
                PRIJAVA privremeni = trenutni->prijava;
                trenutni->prijava = trenutni->sledeci->prijava;
                trenutni->sledeci->prijava = privremeni;
                promena = 1;
            }

            trenutni = trenutni->sledeci;
        }

        prethodni = trenutni;
    } while (promena);
}

```

```

int prover_i_sifru(int sifra_predmeta) {
    int suma = 0;
    int poslednja_cifra = sifra_predmeta % 10;
    sifra_predmeta /= 10;

    while (sifra_predmeta > 0) {
        suma += sifra_predmeta % 10;
        sifra_predmeta /= 10;
    }

    if ((suma % 2 == 0 && poslednja_cifra == 0) || (suma % 2 == 1 &&
poslednja_cifra == 1)) {
        return 1;
    }
    else {
        return 0;
    }
}

```

//POMOCNE FUNKCIJE

```

void prikaziMatricu(int info[][4], int n) {
    if (n == 0) {

```

```

        printf("Matrica je prazna.\n");
        return;
    }

    printf("Matrica:\n");
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < 4; j++) {
            printf("%d ", info[i][j]);
        }
        printf("\n");
    }
}

void prikaziListu(PCVOR glava) {
    if (glava == NULL) {
        printf("Lista je prazna.\n");
        return;
    }

    printf("\nLista:\n");
    PCVOR trenutni = glava;
    int rang = 1;
    while (trenutni != NULL) {
        printf("%d. %d/%d %d %d\n", rang, trenutni->prijava.godina_upisa,
trenutni->prijava.broj_indeksa,
            trenutni->prijava.sifra_predmeta, trenutni->prijava.ocena);
        trenutni = trenutni->sledeci;
        rang++;
    }
}

int main() {
    int info[100][4];
    int n;

    //1. Ucitavanje podataka u matricu
    procitaj_datoteku("ispiti.txt", info, &n);
    prikaziMatricu(info, n);

    //2. Prosečna ocena na predmetu
    double prosečna_ocena = prosečna_ocena_predmet(1113, info, n);
    printf("\nProsečna ocena za predmet sa sifrom 1113: %.21f\n\n",
prosečna_ocena);

    //3. Prikaz studenata koji su položili predmet
    prikazi(1113, info, n);

    //4. Kreiranje liste
    PCVOR glava = NULL;
    kreiraj_listu(info, 2019, &glava, n);
    prikaziListu(glava);

    //5. Sortiranje liste
    sortiraj(glava);
    prikaziListu(glava);

    //7. Provera sifre
    int sifra = 468720;
    int validna_sifra = prover_i_sifru(sifra);
    if (validna_sifra == 1) {

```

```
        printf("\nSifra %d je u dobrom formatu.\n", sifra);
    }
    else {
        printf("\nSifra %d nije u dobrom formatu.\n", sifra);
    }

    return 0;
}
```



## ПРОГРАМИРАЊЕ I

### Практични део испита

Име и презиме: \_\_\_\_\_

Бр. индекса: \_\_\_\_\_

Поени: \_\_\_\_\_

#### Обавештења:

- Испит траје 120 минута.
- На локалном диску С направити фолдер под називом FEBRUAR\_P1T1.
- НАЗИВ ФАЈЛА **ИЗВОРНОГ КОДА** МОРА БИТИ У СЛЕДЕЋЕМ ФОРМАТУ име\_презиме\_годинауписа\_бројиндекса (пример: Pera\_Peric\_2021\_0001.c). Сви задаци се раде у једном пројекту, у једној .c датотеци.
- За сваки задатак где је дат прототип функције мора да се имплементира функција која одговара прототипу (ИСТИ назив, листа параметара и повратна вредност), исто важи и за дате типове.
- Могу се дописивати додатне помоћне функције и типови.

#### Задатак:

Дата је матрица која представља продају производа по данима.

У првој колони се налази шифра производа, док у наредних пет се налазе подаци о продаји производа за пет радних дана. Колоне са индексима 1, 2, 3, 4 и 5 представљају продају производа у понедељак, уторак, среду, четвртак и петак, респективно.

Шифре производа, односно недељне продаје се могу понављати у матрици.

Дата је матрица:

4974 5 10 12 0 1

211112 10 0 4 5 8

63158 0 3 1 0

4974 11 1 6 0 1

63158 0 13 1 7

2134 1 0 4 2 1

Познато је да матрица увек има тачно 6 колона, док је број редова произвољан.

Имплементирати следеће функције:

▪ `int max_prodaja(int info[][6], int n)`

10п.

Функција која враћа шифру производа који је имао највећу недељну продају.

За горе дат пример функција треба да врати шифру 4974 јер је овај производ имао највећу продају ( $5+10+12+0+1=28$ ).

▪ `double prosek(char dan[], int info[][6], int n)`

10п.

Израчунати просек продаје за прослеђени дан. Могуће вредности за параметар дан су:

*pon, uto, sre, cet, pet.*

На пример, за прослеђен дан *uto* и изнад дату матрицу, функција треба да врати ( $10+0+3+1+13+0 = 27 / 6 =$ ) 4.5.

Дати су следећи типови:

```
typedef struct cvor CVOR;
typedef struct cvor * PCVOR;
typedef struct prodaja {
    int sifra_proizvoda;
    int prodata_kolicina;
} PRODAJA;
struct cvor{
    PRODAJA prodaja;
    PCVOR sledeci;
};
```

▪ `void dodaj(PCVOR * glava, PRODAJA prodaja)`

15п.

Функција додаје нови чвор на крај листе.

(Уколико се имплементира додавање на почетак листе будује се са 5п.)



- 
- **void kreiraj\_listu(int info[][6], int n, PCVOR \* glava)** **10п.**  
Функција пребацује податке о укупној недељној продаји сваког производа. Сваки ред у матрици треба превести у један чвор (искористити функцију *dodaj()*) тако да шифра производа одговара првој вредности у реду матрице, док збир продаја свих радних дана представља укупну недељну продају. (Напомена: један ред у матрици одговара једном чвору, што значи да се и у листи могу понављати шифре производа).
  - **void sortiraj(PCVOR glava)** **15п.**  
Функција сортира листу по продатој количини у растућем редоследу.
  - **void izmeni(int sifra, int kolicina, PCVOR glava)** **10п.**  
Функција проналази производ са прослеђеном шифром и мења тренутну количину прослеђеном.
  - **void sacuvaj(int kolicina, char \* naziv, PCVOR glava)** **15п.**  
Направити текстуалну датотеку у коју треба уписати податке о производима који су имали укупну продају већу од задате количине.  
За горе дат пример и количину 10, садржај датотеке треба да буде следећи:  
4974,28  
211112,27  
4974,19  
63158,21
  - **int prover\_i\_sifru(int sifra)** **15п.**  
Функција проверава да ли је шифра у добром формату, уколико јесте враћа 1, а уколико није онда треба да врати 0. Шифра мора имати више од три цифре и мање од 7, а у добром је формату уколико су све цифре, осим прве и последње непарне.  
Пример добрих шифара:  
211112  
63158  
4974



```

#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include <math.h>

//FEBRUAR 2023 - GRUPA 1

typedef struct prodaja {
    int sifra_proizvoda;
    int prodata_kolicina;
} PRODAJA;

typedef struct cvor* PCVOR;

typedef struct cvor {
    PRODAJA prodaja;
    PCVOR sledeci;
} CVOR;

//GLAVNE FUNKCIJE

int max_prodaja(int info[][6], int n) {
    int max_sifra = info[0][0];
    int max_prodaja = info[0][1] + info[0][2] + info[0][3] + info[0][4] +
info[0][5];

    for (int i = 1; i < n; i++) {
        int trenutna_prodaja = info[i][1] + info[i][2] + info[i][3] +
info[i][4] + info[i][5];
        if (trenutna_prodaja > max_prodaja) {
            max_prodaja = trenutna_prodaja;
            max_sifra = info[i][0];
        }
    }

    return max_sifra;
}

double prosek(char dan[], int info[][6], int n) {
    int dan_indeks;
    if (strcmp(dan, "pon") == 0) {
        dan_indeks = 1;
    }
    else if (strcmp(dan, "uto") == 0) {
        dan_indeks = 2;
    }
    else if (strcmp(dan, "sre") == 0) {
        dan_indeks = 3;
    }
    else if (strcmp(dan, "cet") == 0) {
        dan_indeks = 4;
    }
    else if (strcmp(dan, "pet") == 0) {
        dan_indeks = 5;
    }
    else {
        printf("Pogresan dan.\n");
        return 0;
    }
}

```

```

    int suma = 0;
    for (int i = 0; i < n; i++) {
        suma += info[i][dan_indeks];
    }

    return (double)suma / n;
}

void dodaj(PCVOR* glava, PRODAJA prodaja) {
    PCVOR noviCvor = malloc(sizeof(CVOR));
    noviCvor->prodaja = prodaja;
    noviCvor->sledeci = NULL;

    if (*glava == NULL) {
        *glava = noviCvor;
    }
    else {
        PCVOR trenutni = *glava;
        while (trenutni->sledeci != NULL) {
            trenutni = trenutni->sledeci;
        }
        trenutni->sledeci = noviCvor;
    }
}

void kreiraj_listu(int info[][6], int n, PCVOR* glava) {
    for (int i = 0; i < n; i++) {
        PRODAJA prodaja;
        prodaja.sifra_proizvoda = info[i][0];
        prodaja.prodata_kolicina = info[i][1] + info[i][2] + info[i][3] +
info[i][4] + info[i][5];
        dodaj(glava, prodaja);
    }
}

void sortiraj(PCVOR glava) {
    int promena;
    PCVOR trenutni;

    do {
        promena = 0;
        trenutni = glava;

        while (trenutni->sledeci != NULL) {
            if (trenutni->prodaja.prodata_kolicina > trenutni->sledeci-
>prodaja.prodata_kolicina) {
                PRODAJA privremeni = trenutni->prodaja;
                trenutni->prodaja = trenutni->sledeci->prodaja;
                trenutni->sledeci->prodaja = privremeni;
                promena = 1;
            }
            trenutni = trenutni->sledeci;
        }
    } while (promena);
}

void izmeni(int sifra, int kolicina, PCVOR glava) {
    PCVOR trenutni = glava;
    while (trenutni != NULL) {

```

```

        if (trenutni->prodaja.sifra_proizvoda == sifra) {
            trenutni->prodaja.prodata_kolicina = kolicina;
        }
        trenutni = trenutni->sledeci;
    }
}

void sacuvaj(int kolicina, char* naziv, PCVOR glava) {
    FILE* file = fopen(naziv, "w");
    if (file == NULL) {
        printf("Greska pri otvaranju datoteke.\n");
        return;
    }

    PCVOR trenutni = glava;
    while (trenutni != NULL) {
        if (trenutni->prodaja.prodata_kolicina > kolicina) {
            fprintf(file, "%d,%d\n", trenutni->prodaja.sifra_proizvoda,
trenutni->prodaja.prodata_kolicina);
        }
        trenutni = trenutni->sledeci;
    }

    fclose(file);
}

int prover_i_sifru(int sifra) {
    char sifra_str[10];
    sprintf(sifra_str, "%d", sifra);
    int duzina = strlen(sifra_str);
    if (duzina <= 3 || duzina >= 7) {
        return 0;
    }
    for (int i = 1; i < duzina - 1; i++) {
        //ASCII: 0:48; 1:49; 2:50; 3:51; 4:52; 5:53; 6:54; 7:55; 8:56;
9:57
        if ((sifra_str[i] - '0') % 2 != 1) { //sifra_str[1] -> 1
            return 0;
        }
    }

    if ((sifra_str[0] - '0') % 2 != 0 && (sifra_str[duzina] - '0') % 2 !=
0) {
        return 0;
    }

    return 1;
}

int main() {
    int info[][6] = {
        {4974, 5, 10, 12, 0, 1},
        {211112, 10, 0, 4, 5, 8},
        {63158, 0, 3, 1, 0, 0},
        {4974, 11, 1, 6, 0, 1},
        {63158, 0, 13, 1, 7, 0},
        {2134, 1, 0, 4, 2, 1}
    };
    int n = 6;

```

```

PCVOR glava = NULL;

//1. Kreiranje liste
kreiraj_listu(info, n, &glava);

//2. Najveca nedeljna prodaja
int max_sifra = max_prodaja(info, n);
printf("Sifra proizvoda sa najvecom nedeljnom prodajom: %d\n\n",
max_sifra);

//3. Prosek prodaje
double prosek_prodaje = prosek("uto", info, n);
printf("Prosek prodaje za uto: %.2lf\n\n", prosek_prodaje);

//4. Sortirana lista
sortiraj(glava);
printf("Sortirana lista:\n");
PCVOR trenutni = glava;
while (trenutni != NULL) {
    printf("%d, %d\n", trenutni->prodaja.sifra_proizvoda, trenutni-
>prodaja.prodata_kolicina);
    trenutni = trenutni->sledeci;
}

//5. Izmena kolicine
izmeni(63158, 20, glava);
printf("\nLista nakon izmene:\n");
trenutni = glava;
while (trenutni != NULL) {
    printf("%d, %d\n", trenutni->prodaja.sifra_proizvoda, trenutni-
>prodaja.prodata_kolicina);
    trenutni = trenutni->sledeci;
}

//6. Kreiranje izvestaja
sacuvaj(10, "rezultat.txt", glava);
printf("\nPodaci su sacuvani u datoteku rezultat.txt.\n\n");

//7. Provera sifre
int validna_sifra = 211112;
if (proveri_sifru(validna_sifra)) {
    printf("Sifra %d je u dobrom formatu.\n", validna_sifra);
}
else {
    printf("Sifra %d nije u dobrom formatu.\n", validna_sifra);
}

validna_sifra = 63158;
if (proveri_sifru(validna_sifra)) {
    printf("Sifra %d je u dobrom formatu.\n", validna_sifra);
}
else {
    printf("Sifra %d nije u dobrom formatu.\n", validna_sifra);
}

int nevalidna_sifra = 123315;
if (proveri_sifru(nevalidna_sifra)) {
    printf("Sifra %d je u dobrom formatu.\n", nevalidna_sifra);
}
else {

```

```
        printf("Sifra %d nije u dobrom formatu.\n", nevalidna_sifra);  
    }  
    return 0;  
}
```



## ПРОГРАМИРАЊЕ I

### Практични део испита

Име и презиме: \_\_\_\_\_

Бр. индекса: \_\_\_\_\_

Поени: \_\_\_\_\_

#### Обавештења:

- Испит траје 120 минута.
- На локалном диску С направити фолдер под називом FEBRUAR\_P1T2.
- НАЗИВ ФАЈЛА **ИЗВОРНОГ КОДА** МОРА БИТИ У СЛЕДЕЋЕМ ФОРМАТУ име\_презиме\_годинауписа\_бројиндекса (пример: Pera\_Peric\_2021\_0001.c). Сви задаци се раде у једном пројекту, у једној .c датотеци.
- За сваки задатак где је дат прототип функције мора да се имплементира функција која одговара прототипу (ИСТИ назив, листа параметара и повратна вредност), исто важи и за дате типове.
- Могу се дописивати додатне помоћне функције и типови.

#### Задатак:

Потребно је имплементирати следеће функције:

Дата је матрица која садржи податке о поенима које су студенти остварили на практичном и усменом делу испита.

Пример матрице:

20180018 99 12 1

20200001 52 76 2

20100005 78 66 1

Значење вредности по колонама матрице је следеће:

Прва колона – индекс студента,

Друга колона – број освојених поена на практичном делу испита,

Трећа колона – број освојених поена на усменом делу испита,

Четврта колона – шифра испитног рока (1 – јануарски, 2 – фебруарски ...).

Познато је да матрица увек има тачно четири колоне, док број редова може бити произвољан.

Имплементирати следеће функције:

▪ **int ocena(int broj\_indeksa, int rok, int info[][4], int n)** 10п.

Функција рачуна и враћа оцену коју је студент са прослеђеним бројем индекса у задатом року добио на испиту. Да би студент положио предмет потребно је да на оба дела испита остварио више од 50 поена.

Оцена се рачуна тако што се израчуна просек поена које је студент остварио на практичном и усменом делу испита. Уколико је студент остварио између 51 и 60 поена, добија оцену 6, од 61 до 70 оцену 7, од 71 до 80 оцену 8, од 81 до 90 оцену 9, а уколико је остварио више од 90 поена добија оцену 10.

▪ **double prosek(char deo\_ispita, int rok, int info[][4], int n)** 10п.

Прослеђена вредност део испита може да има вредност **p** – писмени или **u** – усмени, у зависности од прослеђене вредности потребно је израчунати просечан број остварених поена на практичном, односно усменом делу испита.

Дати су следећи типови:

```
typedef struct cvor CVOR;
typedef struct cvor * PCVOR;
typedef struct prijava {
    int broj_indeksa;
    int poeni_prakticni;
    int poeni_usmeni;
    int rok;
} PRIJAVA;
struct cvor{
    PRIJAVA prijava;
    PCVOR sledeci;
};
```



- 
- **void dodaj(PCVOR \* glava, PRIJAVA p)** **15п.**  
Функција додаје нови чвор на крај листе.  
(Уколико се имплементира додавање на почетак листе бодује се са 5п.)
  - **void kreiraj\_listu(int info[][4], int n, PCVOR \* glava)** **10п.**  
Функција пребацује податке о студентима и њиховим поенима. Потребно је пребацити само студенте који су положили оба дела испита.
  - **void sortiraj(PCVOR glava)** **15п.**  
Функција сортира листу по оствареној оцени ((писмени+усмени)/2) у опадајућем редоследу.
  - **void pronadji(int rok, PCVOR glava, PRIJAVA \* prijava)** **10п.**  
Функција проналази и у параметар *prijava* уписује пријаву студента који је остварио највећи број поена на практичном делу испита у року који је прослеђен.
  - **void sacuvaj(int rok, char \* naziv, PCVOR glava)** **15п.**  
Направити текстуалну датотеку у коју треба уписати индексе свих студената који су положили испит у задатом року.  
Нпр. за горе дат пример и прослеђен рок 1, датотека треба да буде у следећем формату:  
20180018  
20100005
  - **int prover\_i\_indeks(int indeks)** **15п.**  
Функција проверава да ли је број индекса у добром формату, уколико јесте функције враћа 1, уколико није враћа 0.  
Индекс је у добром формату уколико прве четири цифре одговарају години уписа која може бити између 2000 и 2022, док преостале четири цифре представљају број индекса који може бити између 1 и 999.

```

#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include <math.h>

//FEBRUAR 2023 - GRUPA 2

typedef struct prijava {
    int broj_indeksa;
    int poeni_prakticni;
    int poeni_usmeni;
    int rok;
} PRIJAVA;

typedef struct cvor {
    PRIJAVA prijava;
    struct cvor* sledeci;
} CVOR;

typedef struct cvor* PCVOR;

//GLAVNE FUNKCIJE

int ocena(int broj_indeksa, int rok, int info[][4], int n) {
    int poeni_prakticni = 0;
    int poeni_usmeni = 0;
    int broj_studenata = 0;

    for (int i = 0; i < n; i++) {
        if (info[i][0] == broj_indeksa && info[i][3] == rok) {
            poeni_prakticni = info[i][1];
            poeni_usmeni = info[i][2];
            broj_studenata++;
            break;
        }
    }

    if (broj_studenata == 0 || poeni_prakticni < 51 || poeni_usmeni < 51)
        return 5;

    double prosek = (poeni_prakticni + poeni_usmeni) / 2.0;

    if (prosek > 90 && prosek <= 100)
        return 10;
    else if (prosek > 80)
        return 9;
    else if (prosek > 70)
        return 8;
    else if (prosek > 60)
        return 7;
    else if (prosek > 50)
        return 6;
    else
        return 5;
}

double prosek(char deo_ispita, int rok, int info[][4], int n) {
    int broj_poena = 0;
    int broj_studenata = 0;

```



```

    for (int i = 0; i < n; i++) {
        if (info[i][3] == rok) {
            if (deo_ispita == 'p')
                broj_poena += info[i][1];
            else if (deo_ispita == 'u')
                broj_poena += info[i][2];

            broj_studenata++;
        }
    }

    if (broj_studenata == 0)
        return 0.0;

    return (double)broj_poena / broj_studenata;
}

void dodaj(PCVOR* glava, PRIJAVA p) {
    PCVOR noviCvor = malloc(sizeof(CVOR));
    noviCvor->prijava = p;
    noviCvor->sledeci = NULL;

    if (*glava == NULL) {
        *glava = noviCvor;
        return;
    }

    PCVOR trenutni = *glava;
    while (trenutni->sledeci != NULL) {
        trenutni = trenutni->sledeci;
    }

    trenutni->sledeci = noviCvor;
}

void kreiraj_listu(int info[][4], int n, PCVOR* glava) {
    int i;

    for (i = 0; i < n; i++) {
        int broj_indeksa = info[i][0];
        int poeni_prakticni = info[i][1];
        int poeni_usmeni = info[i][2];
        int rok = info[i][3];

        if (poeni_prakticni > 50 && poeni_usmeni > 50) {
            PRIJAVA nova_prijava;
            nova_prijava.broj_indeksa = broj_indeksa;
            nova_prijava.poeni_prakticni = poeni_prakticni;
            nova_prijava.poeni_usmeni = poeni_usmeni;
            nova_prijava.rok = rok;

            dodaj(glava, nova_prijava);
        }
    }
}

void sortiraj(PCVOR glava) {
    int promena;
    PCVOR trenutni;

```

```

do {
    promena = 0;
    trenutni = glava;

    while (trenutni->sledeci != NULL) {
        double prosek_trenutni = (trenutni->prijava.poeni_prakticni +
trenutni->prijava.poeni_usmeni) / 2.0;
        double prosek_sledeci = (trenutni->sledeci->prijava.poeni_prakticni + trenutni->sledeci->prijava.poeni_usmeni) /
2.0;

        if (prosek_trenutni < prosek_sledeci) {
            PRIJAVA temp = trenutni->prijava;
            trenutni->prijava = trenutni->sledeci->prijava;
            trenutni->sledeci->prijava = temp;

            promena = 1;
        }
        trenutni = trenutni->sledeci;
    }
} while (promena);
}

void pronadji(int rok, PCVOR glava, PRIJAVA* prijava) {
    PCVOR trenutni = glava;
    int max_poeni = 0;

    while (trenutni != NULL) {
        if (trenutni->prijava.rok == rok && trenutni->prijava.poeni_prakticni > max_poeni) {
            max_poeni = trenutni->prijava.poeni_prakticni;
            *prijava = trenutni->prijava;
        }

        trenutni = trenutni->sledeci;
    }
}

void sacuvaj(int rok, char* naziv, PCVOR glava) {
    FILE* file = fopen(naziv, "w");

    if (file == NULL) {
        printf("Greska pri otvaranju datoteke.\n");
        return;
    }

    PCVOR trenutni = glava;

    while (trenutni != NULL) {
        if (trenutni->prijava.rok == rok) {
            fprintf(file, "%d\n", trenutni->prijava.broj_indeksa);
        }

        trenutni = trenutni->sledeci;
    }

    fclose(file);
}

```

```

int prover_i_indeks(int indeks) {
    int godina = indeks / 10000;
    int broj = indeks % 10000;

    if (godina >= 2000 && godina <= 2022 && broj >= 1 && broj <= 999) {
        return 1;
    }

    return 0;
}

//POMOCNE FUNKCIJE

void prikazi_listu(PCVOR glava) {
    PCVOR trenutni = glava;

    if (trenutni == NULL) {
        printf("Lista je prazna.\n");
        return;
    }

    printf("Lista:\n");
    while (trenutni != NULL) {
        printf("Indeks: %d, Poeni prakticni: %d, Poeni usmeni: %d, Rok: %d\n",
            trenutni->prijava.broj_indeksa, trenutni->prijava.poeni_prakticni,
            trenutni->prijava.poeni_usmeni, trenutni->prijava.rok);
        trenutni = trenutni->sledeci;
    }
}

int main() {
    int info[][4] = {
        {20180018, 99, 12, 1},
        {20200001, 52, 76, 2},
        {20100005, 78, 66, 1}
    };
    int n = 3;
    PCVOR glava = NULL;

    //1. Kreiranje liste
    kreiraj_listu(info, n, &glava);
    prikazi_listu(glava);

    //2. Ocena studenta
    int indeks = 20100005;
    int rok = 1;
    int rezultat = ocena(indeks, rok, info, n);
    printf("\nOcena za studenta sa indeksom %d u roku %d: %d\n", indeks, rok, rezultat);

    //3. Prosek poena
    char deo_ispita = 'p';
    double prosečni_poeni = prosek(deo_ispita, rok, info, n);
    printf("\nProsečni poeni na prakticnom delu ispita u roku %d: %.2lf\n\n", rok, prosečni_poeni);

    //4. Sortiraj listu
    sortiraj(glava);
}

```

```

prikazi_listu(glava);

//5. Najbolja prijava
PRIJAVA najbolja_prijava;
pronadji(rok, glava, &najbolja_prijava);
printf("\nNajbolji broj poena na prakticnom delu ispita u roku %d:
%d\n", rok, najbolja_prijava.poeni_prakticni);

//6. Kreiranje izvestaja
sacuvaj(rok, "polozeni.txt", glava);

//7. Provera indeksa
int validan_indeks = 20210001;
int rezultat_provere_validan = prover_i_indeks(validan_indeks);
if (rezultat_provere_validan) {
    printf("\nIndeks %d je u ispravnom formatu.\n", validan_indeks);
}
else {
    printf("Indeks %d nije u ispravnom formatu.\n", validan_indeks);
}

int nevalidan_indeks = 20330000;
int rezultat_provere_nevalidan = prover_i_indeks(nevalidan_indeks);
if (rezultat_provere_nevalidan) {
    printf("Indeks %d je u ispravnom formatu.\n", nevalidan_indeks);
}
else {
    printf("Indeks %d nije u ispravnom formatu.\n",
nevalidan_indeks);
}
return 0;
}

```



## ПРИНЦИПИ ПРОГРАМИРАЊА (ПРОГРАМИРАЊЕ I)

### Практични део испита

Име и презиме: \_

Бр. индекса: \_

Поени: \_

#### Обавештења:

- Испит траје 120 минута.
- На локалном диску С направити фолдер под називом JUN\_T1G1.
- НАЗИВ ФАЈЛА **ИЗВОРНОГ КОДА** МОРА БИТИ У СЛЕДЕЋЕМ ФОРМАТУ име\_презиме\_годинауписа\_бројиндекса (пример: Pera\_Peric\_2021\_0001.c). Сви задаци се раде у једном пројекту, у једној .c датотеци.
- За сваки задатак где је дат прототип функције мора да се имплементира функција која одговара прототипу (ИСТИ назив, листа параметара и повратна вредност), исто важи и за дате типове.
- Могу се дописивати додатне помоћне функције и типови.

#### Задатак:

Дата је следећа структура која представља саобраћајницу:

```
typedef struct saobracajnica{  
    char naziv[50];  
    int id;  
} SAOBRACAJNICA;
```

Имплементирати следеће функције:

#### ▪ SAOBRACAJNICA unesi\_saobracajnicu()

15п.

Омогућити кориснику да унесе податке о саобраћајници. Водити рачуна о томе да *id* може да има вредност између 0 и 99, док *naziv* мора да садржи најмање три карактера.

Уколико корисник погрешно унесе ид и/или назив, потребно је исписати *Pogresan id!* и/или *Pogresan naziv!* Функција враћа структуру саобраћајница са унетим подацима.

```
Unesite id:  
1001  
Unesite naziv:  
Bulevar oslobodjenja  
Pogresan id! Pokusajte ponovo.  
Unesite id:  
1  
Unesite naziv:  
B  
Pogresan naziv! Pokusajte ponovo.  
Unesite id:  
1000  
Unesite naziv:  
B  
Pogresan id! Pogresan naziv! Pokusajte ponovo  
Unesite id:  
0  
Unesite naziv:  
Bulevar oslobodjenja  
Dobar unos!
```

#### ▪ void dodaj\_u\_niz(SAOBRACAJNICA s, SAOBRACAJNICA niz[], int \* n);

10п.

Додаје саобраћајницу на почетак низа.

(Напомена: додавање на крај вреди 5 од 10 поена)

#### ▪ int vrati\_najoptereceniju(int stanje[], int max\_indeks)

15п.

Низ *stanje* представља број возила који се налази на саобраћајницама. Индекс елемента низа представља идентификатор саобраћајнице. Број *max\_indeks* представља максимални капацитет низа *stanje*.

Функција треба да врати идентификатор најоптерећеније саобраћајнице, односно саобраћајнице са највећим бројем возила.



- **void prikazi\_stanje(int stanje[], SAOBRACAJNICA niz[], int n)** **20п.**

Приказати стање на свим саобраћајницама. Потребно је приказати назив саобраћајнице и тренутни број возила. Индекси низа *stanje* одговарају идентификаторима саобраћајница.

На пример, у низу саобраћајница могу да постоје саобраћајнице са индексима 0, 1, 3 и 5. Такође, низ са саобраћајницама није сортиран, тако да могу ићи редоследом 1,3,0,5.

Потребно је приказати стање само за саобраћајнице које се налазе у том низу, сортирано од најмањег до највећег идентификатора, али не сме се сортирати низ саобраћајница, и то на следећи начин:

1 Bulevar oslobođenja 20  
2 Jove Ilica 12

Дате су следеће структуре:

```
typedef struct prekrasaj{
    char registracija[11];
    char tip_prekrasaja[30];
    int id_saobracajnice;
} PREKRSAJ;
typedef struct cvor CVOR;
typedef struct cvor * PCVOR;
struct cvor{
    PREKRSAJ prekrasaj;
    PCVOR sledeci;
}
```

- **void dodaj\_prekrasaj(PREKRSAJ p, PCVOR \* glava)** **10п.**

Функција омогућава додавање новог прекршаја у листу прекршаја, и то на крај листе.

(Напомена: додавање на почетак листе вреди 5 од 10 поена).

- **int broj\_prekrasaja(PCVOR glava, int id\_saobracajnice)** **15п.**

Функција враћа укупан број прекршаја на саобраћајници са прослеђеним идентификатором.

- **void kreiraj\_izvestaj(char \* naziv\_datoteke, PCVOR glava)** **15п.**

Функција уписује податке о прекршајима у датотеку са прослеђеним називом.

У датотеку је неопходно уписати назив саобраћајнице и укупан број прекршаја који је направљен на тој саобраћајници.

4: 20  
1: 0  
2: 12

- **int prover\_i\_registraciju(char registracija[])** **20п.**

Имплементирати функцију која проверава да ли је прослеђена регистрација у одговарајућем формату. Уколико јесте функција треба да врати 1, уколико није треба да врати 0.

Регистрација је одговарајућем формату уколико су прва два карактера велика слова, наредна 4 или 5 бројеви, након тога следи црта, док су последња два карактера велика слова.

Уколико регистрација садржи 5 бројева, онда мора да се завршава карактерима TX.

Добар формат:

BG1111-DL

BG12345-TX

Лош формат:

BGD123456-89

BE01234-AA

B01234-A5

BG12345-AA

```

#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include <math.h>
#define MAX_SAOBRACAJNICE 100

//JUN 2023 - GRUPA 1

typedef struct saobracajnica {
    char naziv[50];
    int id;
} SAOBRACAJNICA;

typedef struct prekrasaj {
    char registracija[11];
    char tip_prekrasaja[30];
    int id_saobracajnice;
} PREKRSAJ;

typedef struct cvor CVOR;
typedef struct cvor* PCVOR;

struct cvor {
    PREKRSAJ prekrasaj;
    PCVOR sledeci;
};

//GLAVNE FUNKCIJE

SAOBRACAJNICA unesi_saobracajnicu() {
    SAOBRACAJNICA s;
    int flag_id;
    int flag_naziv;

    while (1) {
        flag_id = 1;
        flag_naziv = 1;

        printf("Unesite id: ");
        scanf("%d", &s.id);
        getchar();
        if (s.id < 0 || s.id > 99) {
            flag_id = 0;
        }
        printf("Unesite naziv: ");
        fgets(s.naziv, 50, stdin);
        s.naziv[strcspn(s.naziv, "\n")] = '\0';
        if (strlen(s.naziv) < 3) {
            flag_naziv = 0;
        }

        if (flag_id == 1 && flag_naziv == 1) {
            printf("Dobar unos!\n");
            return s;
        }
        else if (flag_id == 0 && flag_naziv == 0) {
            printf("Pogresan id! Pogresan naziv! Pokusajte ponovo.\n");
            continue;
        }
    }
}

```

```

        else if (flag_id == 0) {
            printf("Pogresan id! Pokusajte ponovo.\n");
            continue;
        }
        else {
            printf("Pogresan naziv! Pokusajte ponovo.\n");
            continue;
        }
    }
}

void dodaj_u_niz(SAOBRACAJNICA s, SAOBRACAJNICA niz[], int* n) {

    if (*n == MAX_SAOBRACAJNICE) {
        printf("Niz je popunjen\n");
        return;
    }

    if (*n < MAX_SAOBRACAJNICE) {
        for (int i = *n; i > 0; i--) { // Pomeranje elemenata niza u
desno
            niz[i] = niz[i - 1];
        }
        niz[0] = s;
        (*n)++;
    }
}

int vrati_najoptereceniju(int stanje[], int max_indeks) {
    int najopterecenija = 0;
    int max_vozila = 0;
    for (int i = 0; i < max_indeks; i++) {
        if (stanje[i] > max_vozila) {
            max_vozila = stanje[i];
            najopterecenija = i;
        }
    }
    return najopterecenija;
}

void prikazi_stanje(int stanje[], SAOBRACAJNICA niz[], int n) {

    printf("\nPrikaz saobracajnice sa stanjem: \n");
    for (int i = 0; i < MAX_SAOBRACAJNICE; i++) {
        if (stanje[i] >= 0) {
            for (int j = 0; j < n; j++) {
                if (i == niz[j].id) {
                    printf("%d %s %d\n", i, niz[j].naziv, stanje[i]);
                    break;
                }
            }
        }
    }
}

void dodaj_prekrsaj(PREKRSAJ p, PCVOR* glava) {
    PCVOR noviCvor = malloc(sizeof(CVOR));
    noviCvor->prekrsaj = p;
    noviCvor->sledeci = NULL;
}

```



```

    if (*glava == NULL) {
        *glava = noviCvor;
    }
    else {
        PCVOR trenutni = *glava;
        while (trenutni->sledeci != NULL) {
            trenutni = trenutni->sledeci;
        }
        trenutni->sledeci = noviCvor;
    }
}

int broj_prekrsaja(PCVOR glava, int id_saobracajnice) {
    int broj = 0;
    int flag_id = 0;

    while (glava != NULL) {
        if (glava->prekrsaj.id_saobracajnice == id_saobracajnice) {
            broj++;
            flag_id = 1;
        }
        glava = glava->sledeci;
    }

    if (!flag_id) {
        printf("Nije moguće pronaći %d id u nizu!\n", id_saobracajnice);
        return -1;
    }

    return broj;
}

void kreiraj_izvestaj(char* naziv_datoteke, PCVOR glava) {
    FILE* izvestaj = fopen(naziv_datoteke, "w");
    if (izvestaj == NULL) {
        printf("Greska pri otvaranju datoteke!\n");
        return;
    }

    PCVOR trenutni = glava;
    while (trenutni != NULL) {
        int id = trenutni->prekrsaj.id_saobracajnice;
        int broj = broj_prekrsaja(glava, id);
        fprintf(izvestaj, "%d: %d\n", id, broj);
        trenutni = trenutni->sledeci;
    }

    fclose(izvestaj);
}

int prover_i_registraciju(char registracija[]) {

    int duzina = strlen(registracija);
    //BG1111-DL
    //BG12345-TX

    //Provera duzine registracije
    if (duzina != 9 && duzina != 10) {
        return 0;
    }
}

```

```

    //Provera da li registracija pocinje sa dva velika slova
    if (registracija[0] < 'A' || registracija[0] > 'Z' || registracija[1]
< 'A' || registracija[1] > 'Z') {
        return 0;
    }

    //Provera da li registracija ima cetiri ili pet brojeva
    for (int i = 2; i < duzina - 3; i++) {
        if (registracija[i] < '0' || registracija[i] > '9') {
            return 0;
        }
    }

    //Provera da li se registracija zavrшава sa dva velika slova
    if (registracija[duzina - 2] < 'A' || registracija[duzina - 2] > 'Z'
|| registracija[duzina - 1] < 'A' || registracija[duzina - 1] > 'Z') {
        return 0;
    }

    //Provera da li registracija koja ima 5 brojeva, se zavrшава sa TX
    if (duzina == 10 && registracija[7] == '-' && registracija[8] == 'T'
&& registracija[9] == 'X') {
        return 1;
    }
    else if(duzina == 9 && registracija[6] == '-') {
        return 1;
    }

    return 0;
}

```

//POMOCNE FUNKCIJE

```

void prikazi_saobracajnice(SAOBRACAJNICA niz[], int n) {

    printf("\nNiz saobracajnica:\n");
    for (int i = 0; i < n; i++) {
        printf("id: %2d naziv: %s\n", niz[i].id, niz[i].naziv);
    }
}

void dodaj_stanje(int stanje[], SAOBRACAJNICA niz[], int n) {
    printf("\n");
    int broj;
    int brojac = 0;
    for (int i = 0; i < MAX_SAOBRACAJNICE; i++) {
        for (int j = 0; j < n; j++) {
            if (niz[j].id == i) {
                printf("Uneti broj vozila za [saobracajnica id: %d;
naziv: %s]: ", niz[j].id, niz[j].naziv);
                scanf("%d", &broj);
                getchar();
                stanje[i] = broj;
                break;
            }
            brojac++;
        }
        if (brojac == n) {
            stanje[i] = -1;
        }
    }
}

```

```

    }
}

void ispis_stanja(int stanje[]) {

    printf("\nNiz stanja: \n");
    for (int i = 0; i < MAX_SAOBRACAJNICE; i++) {
        if (stanje[i] >= 0) {
            printf("stanje[%d] = %d\n", i, stanje[i]);
        }
    }
}

void ispis_prekrsaja(PCVOR glava) {

    printf("\nLista prekrsaja:\n");
    while (glava != NULL) {
        printf("id: %2d registracija: %s\n", glava->prekrsaj.id_saobracajnice, glava->prekrsaj.registracija);
        glava = glava->sledeci;
    }
}

int main() {

    //1. Unos saobracajnica
    SAOBRACAJNICA niz[MAX_SAOBRACAJNICE];
    int n = 0;
    for (int i = 0; i < 3; i++) {
        dodaj_u_niz(unesi_saobracajnicu(), niz, &n);
    }
    prikazi_saobracajnice(niz, n);

    //2. Unos stanja na saobracajnicama
    int stanje[MAX_SAOBRACAJNICE];
    dodaj_stanje(stanje, niz, n);
    ispis_stanja(stanje);
    prikazi_stanje(stanje, niz, n);

    //3. Provera najopterećenija saobracajnice
    int index = vrati_najopterećeniju(stanje, MAX_SAOBRACAJNICE);
    printf("\nNajopterećenija saobracajnica je: %d\n\n", stanje[index]);

    //4. Dodavanje 5 prekrsaja
    PREKRSAJ p;
    PCVOR glava = NULL;
    for (int i = 0; i < 5; i++) {
        printf("Unesite registraciju: ");
        scanf("%s", &p.registracija);
        getchar();

        printf("Unesite tip prekrsaja: ");
        fgets(p.tip_prekrsaja, 30, stdin);
        p.tip_prekrsaja[strcspn(p.tip_prekrsaja, "\n")] = '\0';

        printf("Unesite id saobracajnice: ");
        scanf("%d", &p.id_saobracajnice);
        getchar();
    }
}

```

```
        if (!proveri_registraciju(p.registracija)) {
            printf("Pogresan format registracije!\n");
        }
        else {
            dodaj_prekrsaj(p, &glava);
            printf("Prekrsaj dodat!\n");
        }
    }
    ispis_prekrsaja(glava);

    //5. Ukupan broj prekrsaja po saobracajnici
    int id_prekrsaja = 3;
    printf("\nUkupan broj prekrsaja za id %d : %d\n", id_prekrsaja,
    broj_prekrsaja(glava, id_prekrsaja));

    //6. Kreiranje izvestaja
    kreiraj_izvestaj("izvestaj.txt", glava);

    return 0;
}
```



## ПРИНЦИПИ ПРОГРАМИРАЊА (ПРОГРАМИРАЊЕ I)

### Практични део испита

Име и презиме: \_

Бр. индекса: \_

Поени: \_

#### Обавештења:

- Испит траје 120 минута.
- На локалном диску С направити фолдер под називом JUN\_T1G2.
- НАЗИВ ФАЈЛА **ИЗВОРНОГ КОДА** МОРА БИТИ У СЛЕДЕЋЕМ ФОРМАТУ име\_презиме\_годинауписа\_бројиндекса (пример: Pera\_Peric\_2021\_0001.c). Сви задаци се раде у једном пројекту, у једној .c датотеци.
- За сваки задатак где је дат прототип функције мора да се имплементира функција која одговара прототипу (ИСТИ назив, листа параметара и повратна вредност), исто важи и за дате типове.
- Могу се дописивати додатне помоћне функције и типови.

#### Задатак:

Дата је следећа структура која представља саобраћајницу:

```
typedef struct saobracajnica{
    int id; // identifikator
    char naziv[50];
} SAOBRACAJNICA;
```

Имплементирати следеће функције:

**void ucitaj\_saobracajnice(char \* naziv\_datoteke, SAOBRACAJNICA niz[], int \* n)** 15п.

*naziv\_datoteke*: представља назив текстуалне датотеке у којој се налазе подаци о саобраћајницама.

*niz*: низ у који треба уписати све прочитане саобраћајнице из датотеке

*n*: тренутни број чланова низа

Подаци се налазе у следећем формату:

Naziv

Id

Например:

Bulevar oslobodjenja

1

Jove Ilica

4

**void dodaj\_u\_niz(SAOBRACAJNICA s, SAOBRACAJNICA niz[], int \* n);** 15п.

Додаје саобраћајницу на крај низа, без понављања.

(Напомена: додавање са понављањем вреди 5 од 15 поена).

**void postavi\_stanje\_vozila(int broj, int sid, int stanje[], SAOBRACAJNICA niz[], int n)** 20п.

Низ стање представља број возила који се налази на саобраћајницама. Индекс елемента низа представља идентификатор саобраћајнице. Функција треба да постави стање броја возила на саобраћајници. На пример, уколико у низу постоје саобраћајнице са шифрама 0, 1 и 4, онда је могуће мењати стање само елементима на позицијама 0, 1 и 4, док није могуће изменити стање на позицијама 2, 3 итд.

Напомена: низ саобраћајница није сортиран по идентификатору.

**SAOBRACAJNICA prikazi\_namanje\_prometnu(int stanje[], SAOBRACAJNICA niz[], int n)**

**20п.**

Враћа саобраћајницу која је најмање прометна и приказује њен назив у следећем формату:

MIN: Bulevar oslobodjenja

Дате су следеће структуре:

```
typedef struct prekrasaj{
    char tip_prekrasaja[30];
    int id_saobracajnice;
    char registracija[11];
} PREKRASAJ;
```



```
typedef struct cvor CVOR;  
typedef struct cvor * PCVOR;  
struct cvor{  
    PREKRSAJ prekrasaj;  
    PCVOR sledeci;  
}
```

**void dodaj\_prekrasaj(PREKRSAJ p, PCVOR \* glava)**

**10п.**

Функција омогућава додавање новог прекршаја у листу прекршаја, и то на крај листе.

(Напомена: додавање на почетак листе вреди 5 од 10 поена).

**PREKRSAJ unesi\_prekrasaj()**

**10п.**

Омогућава кориснику да унесе податке о новом прекршају у следећем формату:

Unesite tip prekrasaja:  
Prebrza voznja  
Unesite id saobracajnice:  
4  
Unesite registraciju:  
BG1111-DL

**void prikazi(PCVOR glava)**

**10п.**

Функција приказује све податке који се налазе у листи у следећем формату:

Prebrza voznja – 4 – BG1111-DL  
Nevezivanje pojasa – 1 – BG12345-TX

**int prover\_i\_registraciju(char registracija[])**

**20п.**

Имплементирати функцију која проверава да ли је унета регистрација у одговарајућем формату. Уколико јесте функција треба да врати 1, уколико није треба да врати 0.

Регистрација је одговарајућем формату уколико су прва два карактера велика или мала слова, наредна 4 или 5 карактера бројеви, након тога следи црта, док су последња два карактера велика или мала слова.

Добар формат:

**BG1111-DL**

**BG12345-TX**

Лош формат:

**BGD123456-89**

**BE01234-AA**

**B01234-A5**

```

#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include <math.h>
#include <limits.h>
#define MAX_SAOBRACAJNICE 100

//JUN 2023 - GRUPA 2

typedef struct saobracajnica {
    int id; // identifikator
    char naziv[50];
} SAOBRACAJNICA;

typedef struct prekrasaj {
    char tip_prekrasaja[30];
    int id_saobracajnice;
    char registracija[11];
} PREKRSAJ;

typedef struct cvor CVOR;
typedef struct cvor* PCVOR;

struct cvor {
    PREKRSAJ prekrasaj;
    PCVOR sledeci;
};

//GLAVNE FUNKCIJE

void ucitaj_saobracajnice(char* naziv_datoteke, SAOBRACAJNICA niz[], int*
n) {
    FILE* file = fopen(naziv_datoteke, "r");
    if (file == NULL) {
        printf("Greska pri otvaranju datoteke.\n");
        return;
    }

    SAOBRACAJNICA s;
    int i = 0;

    while (fgets(s.naziv, 50, file) != NULL) {
        s.naziv[strcspn(s.naziv, "\n")] = '\0';
        fscanf(file, "%d\n", &s.id);
        niz[i] = s;
        i++;
    }

    *n = i;

    fclose(file);
}

void dodaj_u_niz(SAOBRACAJNICA s, SAOBRACAJNICA niz[], int* n) {
    int i;
    for (i = 0; i < *n; i++) {
        if (niz[i].id == s.id) {
            return;
        }
    }
}

```

```

    }

    niz[*n] = s;
    (*n)++;
}

void postavi_stanje_vozila(int broj, int sid, int stanje[], SAOBRAJNICA
niz[], int n) {

    for (int i = 0; i < MAX_SAOBRACAJNICE; i++){
        if (niz[i].id == sid) {
            stanje[sid] = broj;
            return;
        }
    }
}

SAOBRAJNICA prikazi_najmanje_prometnu(int stanje[], SAOBRAJNICA
niz[], int n) {
    int min_index;
    int min_vozila = INT_MAX;

    for (int i = 0; i < MAX_SAOBRACAJNICE; i++) {
        for (int j = 0; j < n; j++) {
            if (niz[j].id == i) {
                if (stanje[i] < min_vozila) {
                    min_vozila = stanje[i];
                    min_index = i;
                }
            }
        }
    }

    for (int i = 0; i < n; i++) {
        if (niz[i].id == min_index) {
            return niz[i];
        }
    }
}

void dodaj_prekrasaj(PREKRSAJ p, PCVOR* glava) {
    PCVOR noviCvor = malloc(sizeof(CVOR));
    noviCvor->prekrasaj = p;
    noviCvor->sledeci = NULL;

    if (*glava == NULL) {
        *glava = noviCvor;
    }
    else {
        PCVOR trenutni = *glava;
        while (trenutni->sledeci != NULL) {
            trenutni = trenutni->sledeci;
        }
        trenutni->sledeci = noviCvor;
    }
}

PREKRSAJ unesi_prekrasaj() {
    PREKRSAJ p;

```



```

printf("Unesite tip prekrsaja: ");
scanf("%s", p.tip_prekrsaja);
getchar();

printf("Unesite id saobracajnice: ");
scanf("%d", &p.id_saobracajnice);
getchar();

printf("Unesite registraciju: ");
scanf("%s", p.registracija);
getchar();

return p;
}

void prikazi(PCVOR glava) {
    PCVOR trenutni = glava;
    printf("\nLista prekrsaja: \n");
    while (trenutni != NULL) {
        printf("%s - %d - %s\n", trenutni->prekrsaj.tip_prekrsaja,
trenutni->prekrsaj.id_saobracajnice, trenutni->prekrsaj.registracija);
        trenutni = trenutni->sledeci;
    }
}

int prover_i_registraciju(char registracija[]) {

    int duzina = strlen(registracija);
    //BG1111-DL
    //BG12345-TX

    //Provera duzine registracije
    if (duzina != 9 && duzina != 10) {
        return 0;
    }

    //Provera da li registracija pocinje sa dva slova
    if (!isalpha(registracija[0]) || !isalpha(registracija[1])) {
        return 0;
    }

    //Provera da li registracija ima cetiri ili pet brojeva
    for (int i = 2; i < duzina - 3; i++) {
        if (registracija[i] < '0' || registracija[i] > '9') {
            return 0;
        }
    }

    //Provera da li se registracija zavrшава sa dva slova
    if (!isalpha(registracija[duzina - 2]) ||
!isalpha(registracija[duzina - 1])) {
        return 0;
    }

    //Provera da li registracija koja ima 5 brojeva, se završava sa TX
    if (duzina == 10 && registracija[7] == '-' &&
        (registracija[8] == 'T' || registracija[8] == 't') &&
        (registracija[9] == 'X' || registracija[9] == 'x')) {
        return 1;
    }
}

```

```

    }
    else if (duzina == 9 && registracija[6] == '-') {
        return 1;
    }

    return 0;
}

//POMOCNE FUNKCIJE

void prikazi_saobracajnice(SAOBRACAJNICA niz[], int n) {
    for (int i = 0; i < n; i++) {
        printf("id: %2d naziv: %s\n", niz[i].id, niz[i].naziv);
    }
}

void ispis_stanja(int stanje[]) {

    printf("\nNiz stanja: \n");
    for (int i = 0; i < MAX_SAOBRACAJNICE; i++) {
        if (stanje[i] >= 0) {
            printf("stanje[%d] = %d\n", i, stanje[i]);
        }
    }
}

void prikazi_stanje(int stanje[], SAOBRACAJNICA niz[], int n) {

    printf("\nPrikaz saobracajnice sa stanjem: \n");
    for (int i = 0; i < MAX_SAOBRACAJNICE; i++) {
        if (stanje[i] >= 0) {
            for (int j = 0; j < n; j++) {
                if (i == niz[j].id) {
                    printf("%d %s %d\n", i, niz[j].naziv, stanje[i]);
                    break;
                }
            }
        }
    }
}

int main() {

    //1. Ucitavanje saobracajnica
    SAOBRACAJNICA niz[MAX_SAOBRACAJNICE];
    int n = 0;
    ucitaj_saobracajnice("saobracajnice.txt", niz, &n);
    printf("Niz saobracajnica:\n");
    prikazi_saobracajnice(niz, n);

    //2. Dodavanje saobracajnice na kraj niza
    SAOBRACAJNICA nova_saobracajnica;
    nova_saobracajnica.id = 19;
    strcpy(nova_saobracajnica.naziv, "Takovska");
    dodaj_u_niz(nova_saobracajnica, niz, &n);
    printf("\nNovi niz saobracajnica:\n");
    prikazi_saobracajnice(niz, n);

    //3. Dodavanje stanja
    int stanje[MAX_SAOBRACAJNICE];

```

```

for (int i = 0; i < MAX_SAOBRACAJNICE; i++) {
    stanje[i] = -1;
}
postavi_stanje_vozila(54, 91, stanje, niz, n);
postavi_stanje_vozila(62, 63, stanje, niz, n);
postavi_stanje_vozila(52, 42, stanje, niz, n);
postavi_stanje_vozila(38, 22, stanje, niz, n);
postavi_stanje_vozila(73, 19, stanje, niz, n);
postavi_stanje_vozila(29, 12, stanje, niz, n);
ispis_stanja(stanje);
prikazi_stanje(stanje, niz, n);

//4. Najmanje prometna saobracajnica
SAOBRACAJNICA najmanje_prometna = prikazi_najmanje_prometnu(stanje,
niz, n);
printf("\nMIN: %s\n\n", najmanje_prometna.naziv);

//5. Unos prekrasaja
PCVOR glava = NULL;
for (int i = 0; i < 5; i++) {
    PREKRSAJ p = unesi_prekrasaj();

    if (!proveri_registraciju(p.registracija)) {
        printf("Pogresan format registracije!\n");
    }
    else {
        dodaj_prekrasaj(p, &glava);
        printf("Prekrasaj dodat!\n");
    }
}
prikazi(glava);

return 0;
}

```



## ПРИНЦИПИ ПРОГРАМИРАЊА (ПРОГРАМИРАЊЕ I)

### Практични део испита

Име и презиме: \_

Бр. индекса: \_

Поени: \_

#### Обавештења:

- Испит траје 120 минута.
- На локалном диску С направити фолдер под називом JUN\_T2G3.
- НАЗИВ ФАЈЛА **ИЗВОРНОГ КОДА** МОРА БИТИ У СЛЕДЕЋЕМ ФОРМАТУ име\_презиме\_годинауписа\_бројиндекса (пример: Pera\_Peric\_2021\_0001.c). Сви задаци се раде у једном пројекту, у једној .c датотеци.
- За сваки задатак где је дат прототип функције мора да се имплементира функција која одговара прототипу (ИСТИ назив, листа параметара и повратна вредност), исто важи и за дате типове.
- Могу се дописивати додатне помоћне функције и типови.

#### Задатак:

Дата је следећа структура која представља запосленог:

```
typedef struct zaposleni{
    int id;
    char ime[50];
    char prezime[50];
    double cena_rada; //cena rada po satu
} ZAPOSLENI;
```

Имплементирати следеће функције:

**void unos\_zaposlenog(ZAPOSLENI zaposleni[], int \* n)** **10п.**

Функција омогућава кориснику да унесе податке о запосленом и уноси запосленог у низ запослених. Цена рада мора бити већа од 0, име и презиме морају бити дужи од 2 карактера. Уколико корисник погрешно при уносу неког од атрибута, потребно је исписати GRESKA, прекинути унос и не додавати запосленог у низ. Уколико је унос коректан додати новог запосленог на крај низа.

Пример доброг уноса:

```
Unesite id:
0
Unesite ime:
Pera
Unesite prezime:
Peric
Unesite cenu rada po satu:
1250.45
```

Пример лошег уноса:

```
Unesite id:
0
Unesite ime:
P
GRESKA
```

▪ **void sortiraj(ZAPOSLENI zaposleni[], int n)** **20п.**

Функција сортира запослене у опадајућем редоследу, по цени рада.

▪ **void filtriraj\_cene(double cene[], int \* n, double min\_cena, double max\_cena)** **20п.**

Функција избацује из низа све цене рада које нису у прослеђеном рангу.

▪ **void pretrazi\_po\_imenu(char ime[], ZAPOSLENI zaposleni[], int n)** **15п.**

Приказује запослене чије име почиње са *ime*.

(Напомена: уколико се уради да приказује запосленог чије је име једнако прослеђеном стрингу, онда се бодује са 5 од 10п.)

▪ **int pocinje\_sa(char s1[], char s2[])** **15п.**

Функција враћа 1 уколико стринг *s1* почиње карактерима из стринга *s2*, у супротном враћа 0.



(Напомена: забрањена је употреба функције `strncmp`)

Пример:

s1	s2	povratna vrednost
petar	pet	1
petric	petr	1
pera	pera	1
mika	pe	0

Дате су следеће структуре које представљају ангажовање запосленог на задатку.

```
typedef struct angazovanje{
    int id_zaposlenog;
    char naziv_projekta[100];
    int broj_sati;
}ANGAZOVANJE;
typedef struct cvor CVOR;
typedef struct cvor * PCVOR;
struct cvor{
    ANGAZOVANJE angazovanje;
    PCVOR * sledeci
};
```

- `void dodaj_angazovanje(ANGAZOVANJE a, PCVOR * glava)`

10п.

Функција додаје ангажовање на крај листе.

- `double ukupna_primanja(ZAPOSLENI z, PCVOR glava)`

10п.

Враћа колико је укупно зарадио прослеђени запослени.

Укупна примања = цена рада \* укупан број сати ангажовања

- `void ispisi_projekte(char * naziv_datoteke, PCVOR glava)`

20п.

Називи пројеката у листи могу да се понављају, с обзиром на то да може да постоји више од једног ангажовања на једном пројекту. У текстуалну датотеку са прослеђеним називом потребно је уписати називе свих пројеката, без понављања.

(Напомена: исписивање свих пројеката, са понављањем бодује се са 10 од 20 поена).

```

#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include <math.h>
#define MAX 100

//JUN 2023 - GRUPA 3

typedef struct zaposleni {
    int id;
    char ime[50];
    char prezime[50];
    double cena_rada;
} ZAPOSLENI;

typedef struct angazovanje {
    int id_zaposlenog;
    char naziv_projekta[100];
    int broj_sati;
} ANGAZOVANJE;

typedef struct cvor {
    ANGAZOVANJE angazovanje;
    struct cvor* sledeci;
} CVOR;

typedef struct cvor* PCVOR;

//GLAVNE FUNKCIJE

void unos_zaposlenog(ZAPOSLENI zaposleni[], int* n) {

    if (*n == MAX) {
        printf("Niz je popunjen\n");
        return;
    }

    int id;
    char ime[50];
    char prezime[50];
    double cena_rada;

    printf("Unesite id: ");
    scanf("%d", &id);
    getchar();

    printf("Unesite ime: ");
    fgets(ime, sizeof(ime), stdin);
    ime[strcspn(ime, "\n")] = '\0';
    if (strlen(ime) < 3) {
        printf("GRESKA\n");
        return;
    }

    printf("Unesite prezime: ");
    fgets(prezime, sizeof(prezime), stdin);
    prezime[strcspn(prezime, "\n")] = '\0';
    if (strlen(prezime) < 3) {
        printf("GRESKA\n");
    }
}

```

```

        return;
    }

    printf("Unesite cenu rada po satu: ");
    scanf("%lf", &cena_rada);
    getchar();
    if (cena_rada <= 0) {
        printf("GRESKA\n");
        return;
    }
    printf("\n");
    zaposleni[*n].id = id;
    strcpy(zaposleni[*n].ime, ime);
    strcpy(zaposleni[*n].prezime, prezime);
    zaposleni[*n].cena_rada = cena_rada;
    (*n)++;
}

void sortiraj(ZAPOSLANI zaposleni[], int n) {
    for (int i = 0; i < n - 1; i++) {
        for (int j = i + 1; j < n; j++) {
            if (zaposleni[i].cena_rada < zaposleni[j].cena_rada) {
                ZAPOSLENI temp = zaposleni[i];
                zaposleni[i] = zaposleni[j];
                zaposleni[j] = temp;
            }
        }
    }
}

void filtriraj_cene(double cene[], int* n, double min_cena, double
max_cena) {
    int k = 0;
    for (int i = 0; i < *n; i++) {
        if (cene[i] >= min_cena && cene[i] <= max_cena) {
            cene[k] = cene[i];
            k++;
        }
    }
    *n = k;
}

void pretrazi_po_imenu(char ime[], ZAPOSLENI zaposleni[], int n) {
    int flag = 0;
    printf("Zaposleni sa imenom %s:\n", ime);
    for (int i = 0; i < n; i++) {
        if (pocinje_sa(zaposleni[i].ime, ime)) {
            printf("%d. %s %s\n", zaposleni[i].id, zaposleni[i].ime,
zaposleni[i].prezime);
            flag = 1;
        }
    }

    if (!flag) {
        printf("Nije moguće pronaći zaposlenog sa imenom: %s\n", ime);
    }
}

int pocinje_sa(char s1[], char s2[]) {

```

```

    int n = strlen(s2);
    for (int i = 0; i < n; i++) {
        if (s1[i] != s2[i]) {
            return 0;
        }
    }
    return 1;
}

void dodaj_angazovanje(ANGAZOVANJE a, PCVOR* glava) {
    PCVOR noviCvor = malloc(sizeof(CVOR));
    noviCvor->angazovanje = a;
    noviCvor->sledeci = NULL;

    if (*glava == NULL) {
        *glava = noviCvor;
    }
    else {
        PCVOR tekuci = *glava;
        while (tekuci->sledeci != NULL) {
            tekuci = tekuci->sledeci;
        }
        tekuci->sledeci = noviCvor;
    }
}

double ukupna_primanja(ZAPOSLjeni z, PCVOR glava) {
    double ukupna_zarada = 0.0;
    PCVOR tekuci = glava;
    int flag_id = 0;

    while (tekuci != NULL) {
        if (tekuci->angazovanje.id_zaposlenog == z.id) {
            flag_id = 1;
            ukupna_zarada += z.cena_rada * tekuci->angazovanje.broj_sati;
        }
        tekuci = tekuci->sledeci;
    }
    if (!flag_id) {
        printf("Zaposleni sa %d id ne postoji\n", z.id);
        return -1;
    }

    return ukupna_zarada;
}

void ispisi_projekte(char* naziv_datoteke, PCVOR glava) {
    FILE* file = fopen(naziv_datoteke, "w");
    if (file == NULL) {
        printf("Greska pri otvaranju datoteke.\n");
        return;
    }

    PCVOR tekuci = glava;
    while (tekuci != NULL) {
        int vecPostoji = 0;
        PCVOR temp = glava;
        while (temp != tekuci) {
            if (strcmp(temp->angazovanje.naziv_projekta, tekuci->angazovanje.naziv_projekta) == 0) {

```



```

        vecPostoji = 1;
        break;
    }
    temp = temp->sledeci;
}

if (!vecPostoji) {
    fprintf(file, "%s\n", tekuci->angazovanje.naziv_projekta);
}

tekuci = tekuci->sledeci;
}

fclose(file);
}

//POMOCNE FUNKCIJE

void ispis_zaposlenih(ZAPOSLENI zaposleni[], int n) {
    for (int i = 0; i < n; i++) {
        printf("id: %d, ime: %s, prezime: %s, cena rada: %.2lf\n",
            zaposleni[i].id, zaposleni[i].ime, zaposleni[i].prezime,
            zaposleni[i].cena_rada);
    }
}

void ispis_cena(double cene[], int broj_cena) {
    for (int i = 0; i < broj_cena; i++) {
        printf("cena rada: %.2lf\n", cene[i]);
    }
}

void ispis_angazovanja(PCVOR glava) {
    while (glava != NULL) {
        printf("id: %d broj sati: %d naziv projekta: %s\n", glava-
            >angazovanje.id_zaposlenog, glava->angazovanje.broj_sati, glava-
            >angazovanje.naziv_projekta);
        glava = glava->sledeci;
    }
}

int main() {
    //1. Unos zaposlenog
    ZAPOSLENI zaposleni[MAX];
    int n = 0;
    for (int i = 0; i < 5; i++) {
        unos_zaposlenog(zaposleni, &n);
    }
    printf("Zaposleni:\n");
    ispis_zaposlenih(zaposleni, n);

    //2. Sortiranje zaposlenih
    sortiraj(zaposleni, n);
    printf("\nZaposleni nakon sortiranja:\n");
    ispis_zaposlenih(zaposleni, n);

    //3. Filtriranje cena
    double cene[MAX];

```

```

int broj_cena = n;
for (int i = 0; i < n; i++) {
    cene[i] = zaposleni[i].cena_rada;
}
filtriraj_cene(cene, &broj_cena, 1000.0, 2000.0);
ispis_cena(cene, broj_cena);
printf("\n");

//4. Pretraga imena
pretrazi_po_imenu("Petar", zaposleni, n);
pretrazi_po_imenu("Mar", zaposleni, n);
pretrazi_po_imenu("Iml", zaposleni, n);

//5. Angazovanje zaposlenih
PCVOR glava = NULL;
ANGAZOVANJE a1 = { 1, "Projekat A", 40 };
ANGAZOVANJE a2 = { 2, "Projekat V", 30 };
ANGAZOVANJE a3 = { 1, "Projekat C", 20 };
ANGAZOVANJE a4 = { 3, "Projekat B", 10 };
ANGAZOVANJE a5 = { 4, "Projekat U", 50 };
ANGAZOVANJE a6 = { 5, "Projekat B", 70 };
dodaj_angazovanje(a1, &glava);
dodaj_angazovanje(a2, &glava);
dodaj_angazovanje(a3, &glava);
dodaj_angazovanje(a4, &glava);
dodaj_angazovanje(a5, &glava);
dodaj_angazovanje(a6, &glava);
printf("\n");

//6. Ukupna primanja zaposlenih
for (int i = 0; i < n; i++) {
    double ukupna_zarada = ukupna_primanja(zaposleni[i], glava);
    printf("Zaposleni %s %s je zaradio: %.2lf\n", zaposleni[i].ime,
zaposleni[i].prezime, ukupna_zarada);
}
printf("\n");

//7. Prikaz projekata
ispisi_projekte("projekti.txt", glava);
ispis_angazovanja(glava);

return 0;
}

```



## ПРИНЦИПИ ПРОГРАМИРАЊА (ПРОГРАМИРАЊЕ I)

### Практични део испита

Име и презиме: \_

Бр. индекса: \_

Поени: \_

#### Обавештења:

- Испит траје 120 минута.
- На локалном диску С направити фолдер под називом JUN\_T2G4.
- НАЗИВ ФАЈЛА **ИЗВОРНОГ КОДА** МОРА БИТИ У СЛЕДЕЋЕМ ФОРМАТУ име\_презиме\_годинауписа\_бројиндекса (пример: Pera\_Peric\_2021\_0001.c). Сви задаци се раде у једном пројекту, у једној .c датотеци.
- За сваки задатак где је дат прототип функције мора да се имплементира функција која одговара прототипу (ИСТИ назив, листа параметара и повратна вредност), исто важи и за дате типове.
- Могу се дописивати додатне помоћне функције и типови.

#### Задатак:

Дата је следећа структура која представља запосленог:

```
typedef struct zaposleni{
    int id; // identifikator
    char imeiprezime[50];
    double cena_rada; //cena rada po satu
} ZAPOSLENI;
```

Имплементирати следеће функције:

**void unos\_zaposlenog(ZAPOSLENI zaposleni[], int \* n)** **10п.**

Функција омогућава кориснику да унесе податке о запосленом и уноси запосленог у низ запослених на крај низа. Пример уноса:

```
Unesite id:
0
Unesite ime i prezime:
Pera Peric
Unesite cenu rada po satu:
1250.45
```

▪ **double izracunaj\_medijanu(double cene\_rada[], int n)** **20п.**

Функција треба да израчуна и врати медијалну цену рада по сату.

Медијална плата је вредност од које је тачно половина елемената мања, док је друга половина већа од те вредности. Да би се израчунала медијана потребно је прво сортирати низ. Затим, уколико низ има непаран број елемената, вратити вредност средњег, а уколико има паран број елемената вратити просек два елемента у средини.

На пример, за низ цена рада:

100, 100, **200**, 300, 350 медијана је 200 јер је то елемент у средини

100, **100, 300**, 350 медијана је  $(100+300)/2=200$ .

▪ **void izbaci(int id, ZAPOSLENI zaposleni[], int \*n)** **10п.**

Функција треба да избаци из низа запосленог са прослеђеним идентификатором (*id*).

▪ **void pretrazi\_po\_imenu(char pretraga[], ZAPOSLENI zaposleni[], int n)** **15п.**

Приказује запослене чије се име НЕ завршава са *pretraga*.

(Напомена: уколико се уради да приказује запослено чије је име једнако прослеђеном стрингу, онда се бодује са 7 од 15п.)

Нека је дата листа запослених са именима: *Pera Lubarda, Mika Mikic, Zika Sremac, Petar Petronijevic, Ana Anic, Jovana Zec, Pedja Petric*

Уколико се функцији проследи *pretraga ic* на стандардном излазном уређају треба да се прикажу само имена и презимена запослених:

```
Pera Lubarda
Zika Sremac
Jovana Zec
```



- `int završava_sa(char s1[], char s2[])`

20п.

Функција враћа 1 уколико се стринг `s1` завршава карактерима из стринга `s2`, у супротном враћа 0.

(Напомена: забрањена је употреба функције `strncmp`)

Пример:

s1	s2	povratna vrednost
petar	ar	1
petric	tric	1
pera	pera	1
mika	era	0

Дате су следеће структуре које представљају ангажовање запосленог на задатку.

```
typedef struct angazovanje{
    int id_zaposlenog;
    int broj_sati;
    int id_angazovanja; // poslednje dve cifre predstavljaju id projekta
    char naziv_projekta[100];
}ANGAZOVANJE;
typedef struct cvor CVOR;
typedef struct cvor * PCVOR;
struct cvor{
    ANGAZOVANJE angazovanje;
    PCVOR sledeci;
};
```

- `void dodaj_angazovanje(ANGAZOVANJE a, PCVOR * glava)`

10п.

Функција додаје ангажовање на крај листе.

- `double broj_sati_projekta(int id_projekta, PCVOR glava)`

15п.

Враћа укупан број сати ангажовања на прослеђеном пројекту. Познато је да последње две цифре ангажовања представљају шифру пројекта. Потребно је вратити укупан број сати свих ангажовања, чије супоследње две цифре једнаке прослеђеној.

- `void ispisi_projekte(char * naziv_datoteke, PCVOR glava)`

20п.

Називи пројеката у листи могу да се понављају, с обзиром на то да може да постоји више од једног ангажовања на једном пројекту. У текстуалну датотеку са прослеђеним називом потребно је уписати називе свих пројеката, без понављања.

(Напомена: исписивање свих пројеката, са понављањем бодује се са 10 од 20 поена).

```

#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include <math.h>

//JUN 2023 - GRUPA 4

typedef struct zaposleni {
    int id; // identifikator
    char imeiprezime[50];
    double cena_rada; //cena rada po satu
} ZAPOSLENI;

typedef struct angazovanje {
    int id_zaposlenog;
    int broj_sati;
    int id_angazovanja; // poslednje dve cifre predstavljaju id projekta
    char naziv_projekta[100];
} ANGAZOVANJE;

typedef struct cvor CVOR;
typedef struct cvor* PCVOR;

struct cvor {
    ANGAZOVANJE angazovanje;
    PCVOR sledeci;
};

//GLAVNE FUNKCIJE

void unos_zaposlenog(ZAPOSLENI zaposleni[], int* n) {
    printf("Unesite ID: ");
    scanf("%d", &zaposleni[*n].id);
    printf("Unesite ime i prezime: ");
    scanf(" %[^\\n]s", zaposleni[*n].imeiprezime);
    printf("Unesite cenu rada po satu: ");
    scanf("%lf", &zaposleni[*n].cena_rada);

    (*n)++;
}

double izracunaj_medijanu(double cene_rada[], int n) {
    for (int i = 0; i < n - 1; i++) {
        for (int j = 0; j < n - i - 1; j++) {
            if (cene_rada[j] > cene_rada[j + 1]) {
                double temp = cene_rada[j];
                cene_rada[j] = cene_rada[j + 1];
                cene_rada[j + 1] = temp;
            }
        }
    }

    if (n % 2 == 0) {
        return (cene_rada[n / 2] + cene_rada[n / 2 - 1]) / 2.0;
    }
    else {
        return cene_rada[n / 2];
    }
}

```

```

void izbaci(int id, ZAPOSLENI zaposleni[], int* n) {
    for (int i = 0; i < *n; i++) {
        if (zaposleni[i].id == id) {
            for (int j = i; j < *n - 1; j++) {
                zaposleni[j] = zaposleni[j + 1];
            }
            (*n)--;
            printf("Zaposleni sa ID %d je uklonjen.\n", id);
            return;
        }
    }

    printf("Zaposleni sa ID %d nije pronadjen.\n", id);
}

int završava_sa(char s1[], char s2[]) {
    int len1 = strlen(s1);
    int len2 = strlen(s2);

    if (len2 > len1) {
        return 0;
    }

    for (int i = 0; i < len2; i++) {
        if (s1[len1 - len2 + i] != s2[i]) {
            return 0;
        }
    }

    return 1;
}

void pretraži_po_imenu(char pretraga[], ZAPOSLENI zaposleni[], int n) {
    int flag = 0;
    printf("Rezultati pretrage:\n");
    for (int i = 0; i < n; i++) {
        if (!završava_sa(zaposleni[i].imeiprezime, pretraga)) {
            printf("%s\n", zaposleni[i].imeiprezime);
            flag = 1;
        }
    }

    if (!flag) {
        printf("Nije moguće pronaći zaposlenog čije se ime završava sa: %s\n", pretraga);
    }
}

void dodaj_angazovanje(ANGAZOVANJE a, PCVOR* glava) {
    PCVOR noviCvor = malloc(sizeof(CVOR));
    noviCvor->angazovanje = a;
    noviCvor->sledeći = NULL;

    if (*glava == NULL) {
        *glava = noviCvor;
    }
    else {
        PCVOR tekuci = *glava;
        while (tekuci->sledeći != NULL) {

```

```

        tekuci = tekuci->sledeci;
    }
    tekuci->sledeci = noviCvor;
}

double broj_sati_projekta(int id_projekta, PCVOR glava) {
    double ukupno_sati = 0.0;

    PCVOR tekuci = glava;
    while (tekuci != NULL) {
        int poslednje_dve_cifre = tekuci->angazovanje.id_angazovanja %
100;
        if (poslednje_dve_cifre == id_projekta) {
            ukupno_sati += tekuci->angazovanje.broj_sati;
        }
        tekuci = tekuci->sledeci;
    }

    return ukupno_sati;
}

```

```

void ispisi_projekte(char* naziv_datoteke, PCVOR glava) {
    FILE* file = fopen(naziv_datoteke, "w");
    if (file == NULL) {
        printf("Greska pri otvaranju datoteke.\n");
        return;
    }

    PCVOR tekuci = glava;
    while (tekuci != NULL) {
        int vecPostoji = 0;
        PCVOR temp = glava;
        while (temp != tekuci) {
            if (strcmp(temp->angazovanje.naziv_projekta, tekuci-
>angazovanje.naziv_projekta) == 0) {
                vecPostoji = 1;
                break;
            }
            temp = temp->sledeci;
        }

        if (!vecPostoji) {
            fprintf(file, "%s\n", tekuci->angazovanje.naziv_projekta);
        }

        tekuci = tekuci->sledeci;
    }

    fclose(file);
}

```

//POMOCNE FUNKCIJE

```

void ispis_zaposlenih(ZAPOSLjeni zaposleni[], int n) {
    printf("\nZaposleni:\n");
    for (int i = 0; i < n; i++) {
        printf("id: %d, ime i prezime: %s, cena rada: %.2lf\n",
zaposleni[i].id, zaposleni[i].imeiprezime, zaposleni[i].cena_rada);
    }
}

```

```
}
```

```
int main() {
```

```
    //1. Unos zaposlenog
    ZAPOSLENI zaposleni[100] = {
        {1, "Pera Lubarda", 150},
        {2, "Mika Mikic", 200},
        {3, "Zika Sremac", 120},
        {4, "Petar Petronijevic", 80},
        {5, "Ana Anic", 70},
        {6, "Jovana Zec", 140},
    };
    int n = 6;
    unos_zaposlenog(zaposleni, &n);
    ispis_zaposlenih(zaposleni, n);

    //2. Medijalna cena rada
    double cene_rada[100];
    for (int i = 0; i < n; i++) {
        cene_rada[i] = zaposleni[i].cena_rada;
    }
    double medijana = izracunaj_medijanu(cene_rada, n);
    printf("\nMedijana cene rada: %.2lf\n", medijana);

    //3. Izbacivanje zaposlenog iz niza
    int id;
    printf("\nUnesite ID zaposlenog za uklanjanje: ");
    scanf("%d", &id);
    izbaci(id, zaposleni, &n);
    ispis_zaposlenih(zaposleni, n);

    //4. Pretraga po imenu
    char pretraga[50];
    printf("\nUnesite deo imena za pretragu: ");
    scanf("%s", pretraga);
    pretrazi_po_imenu(pretraga, zaposleni, n);

    //5. Angazovanje zaposlenih
    PCVOR glava = NULL;
    ANGAZOVANJE a1 = { 1, 10, 110, "Projekat A" };
    ANGAZOVANJE a2 = { 2, 15, 120, "Projekat B" };
    ANGAZOVANJE a3 = { 1, 8, 130, "Projekat C" };
    ANGAZOVANJE a4 = { 3, 10, 130, "Projekat C" };
    ANGAZOVANJE a5 = { 4, 15, 110, "Projekat A" };
    ANGAZOVANJE a6 = { 5, 8, 140, "Projekat D" };
    dodaj_angazovanje(a1, &glava);
    dodaj_angazovanje(a2, &glava);
    dodaj_angazovanje(a3, &glava);
    dodaj_angazovanje(a4, &glava);
    dodaj_angazovanje(a5, &glava);
    dodaj_angazovanje(a6, &glava);

    //6. Broj sati na projektu
    int id_projekta;
    printf("\nUnesite ID projekta za brojanje sati: ");
    scanf("%d", &id_projekta);
    double broj_sati = broj_sati_projekta(id_projekta, glava);
    printf("Ukupan broj sati na projektu: %.2lf\n", broj_sati);
```



```
//7. Prikaz projekata  
ispisi_projekte("projekti.txt", glava);  
  
return 0;  
}
```



## ПРИНЦИПИ ПРОГРАМИРАЊА (ПРОГРАМИРАЊЕ I)

### Практични део испита

Име и презиме: \_ \_

Бр. индекса: \_

Поени: \_

#### Обавештења:

- Испит траје 120 минута.
- На локалном диску С направити фолдер под називом JUL\_T1G1.
- НАЗИВ ФАЈЛА **ИЗВОРНОГ КОДА** МОРА БИТИ У СЛЕДЕЋЕМ ФОРМАТУ име\_презиме\_годинауписа\_бројиндекса (пример: Pera\_Peric\_2021\_0001.c). Сви задаци се раде у једном пројекту, у једној .c датотеци.
- За сваки задатак где је дат прототип функције мора да се имплементира функција која одговара прототипу (ИСТИ назив, листа параметара и повратна вредност), исто важи и за дате типове.
- Могу се дописивати додатне помоћне функције и типови.

#### Задатак:

Дата је следећа структура која представља посету љубимца ветеринарској ординацији:

```
typedef struct poseta{
    char ime_ljubimca[50];
    char vrsta[35];
    char datum_posete[12]; // datum je u
    formatu dd.MM.yyyy.
} POSETA;

typedef struct cvor CVOR;
typedef struct cvor * PCVOR;
struct cvor{
    char ime_ljubimca[50];
    char vrsta[35];
    PCVOR sledeci;
};
```

Потребно је имплементирати следеће функције:

▪ **int format\_datuma(char datum[])**

**20п.**

Функција проверава да ли је датум правилно форматиран. Уколико јесте правилно форматиран, онда функција треба да врати 1, уколико није треба да врати 0.

За 10 од 20 поена обезбедити да:

Формат датума буде: dd.MM.yyyy.

**dd** представља датум који може имати вредност између 1 и 31,

**MM** представља месец који може имати вредност између 1 и 12

**yyyy** представља годину која може имати вредности између 1950 и 2030.

За укупно 20 поена, неопходно је проверити и да:

Уколико је месец 1,3,5,7,8,10. или 12. онда дан не може имати вредност већу од 31.

Уколико је месец 4, 6, 9, 11. онда дан не може имати вредност већу од 30.

Уколико је месец 2. онда дан не може имати вредност већу од 29.

▪ **void ucitaj\_posete(char \* naziv\_datoteke, POSETA posete[], int \* n)**

**15п.**

Дата је текстуална датотека у коју су уписани подати о посетама кућних љубимаца ветеринару. Подаци о посетама су сортирани по датуму у неоппадајућем редоследу. Подаци се налазе у текстуалној датотеци у следећем формату:

```
ime_ljubimca
vrsta
datum_posete
```

На пример:

```
Garfild
Macka
25.6.2023.
Dzeki
Pas
25.6.2023.
Garfild
Macka
26.6.2023.
```



▪ **void dodaj\_posetu(POSETA p, POSETA posete[], int \* n)** **20п.**

Потребно је омогућити додавање посете у низ, али тако да низ остане сортиран. Низ је сортиран по датуму у неоппадајућем редоследу. Уколико постоји неколико посета истог датума, нову посету коју треба додати за тај датум треба додати на крај подниза.

(за 10 од 20п. додати посету на почетак подниза)

На пример, уколико постоји следећи низ:

0	1	2	3	4	5
Garfild Macka 25.6.2023.	Dzeki Pas 25.6.2023.	Garfild Macka 27.6.2023.	Reks Pas 27.6.2023.	Mini Macka 29.6.2023.	

Уколико треба додати нову посету: Bela,Pas,27.6.2023. низ би требало да изгледа овако

0	1	2	3	4	5
Garfild Macka 25.6.2023.	Dzeki Pas 25.6.2023.	Garfild Macka 27.6.2023.	Reks Pas 27.6.2023.	<b>Bela Pas 27.6.2023.</b>	Reks Macka 29.6.2023.

▪ **void kreiraj\_listu(PCVOR \* glava, POSETA posete[], int n)** **20п.**

Потребно је имплементирати функцију која пребацује податке о свим љубимцима из низа љубимаца у листу. Водити рачуна о томе да у низу посета може да постоји више посета истог љубимца, док подаци о љубимцима не би требало да се понављају.

За изнад дати низ, у листи би требало да се нађу следећи подаци:

Garfild Macka  
Dzeki Pas  
Reks Pas  
Bela Pas  
Reks Macka

▪ **void izbaci\_iz\_liste(PCVOR \* glava, char ime[])** **15п.**

Функција избацује љубимце из листе љубимаца са прослеђеним именом.

▪ **void kreiraj\_matricu(POSETA posete[], int n, int statistika[3][12])** **20п.**

Прослеђена матрица има тачно 12 колона и тачно 3 реда. Потребно је креирати статистику посета. Свака колона представља месец – прва колона представља јануар, друга фебруар итд.

Први ред у матрици представља мачке, други псе, а трећи остале врсте љубимаца. Потребно је уписати у одговарајуће елементе матрице информације о укупном броју посета за сваку врсту животиња у сваком месецу.

▪ **int najposeceniji\_mesec(int statistika[3][12])** **10п.**

Познато је да матрица има тачно 12 колона и тачно 3 реда. Функција треба да врати који месец је имао највише посета. На пример, уколико је фебруар имао највише посета, онда функција треба да врати број 2).

```

#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include <math.h>

//JUL 2023 - GRUPA 1

typedef struct poseta {
    char ime_ljubimca[50];
    char vrsta[35];
    char datum_posete[12]; // datum je u formatu dd.MM.yyyy.
} POSETA;

typedef struct cvor CVOR;

typedef struct cvor* PCVOR;

struct cvor {
    char ime_ljubimca[50];
    char vrsta[35];
    PCVOR sledeci;
};

//GLAVNE FUNKCIJE

int format_datuma(char datum[]) {
    int dd, MM, yyyy;
    sscanf(datum, "%d.%d.%d.", &dd, &MM, &yyyy);

    if (dd < 1 || dd > 31 || MM < 1 || MM > 12 || yyyy < 1950 || yyyy >
2030) {
        return 0;
    }

    if ((MM == 1 || MM == 3 || MM == 5 || MM == 7 || MM == 8 || MM == 10
|| MM == 12) && (dd < 1 || dd > 31)) {
        return 0;
    }

    if ((MM == 4 || MM == 6 || MM == 9 || MM == 11) && (dd < 1 || dd >
30)) {
        return 0;
    }

    if (MM == 2) {
        int prestupna = (yyyy % 4 == 0 && yyyy % 100 != 0) || (yyyy % 400
== 0);
        if ((prestupna && (dd < 1 || dd > 29)) || (!prestupna && (dd < 1
|| dd > 28))) {
            return 0;
        }
    }

    return 1;
}

void ucitaj_posete(char* naziv_datoteke, POSETA posete[], int* n) {
    FILE* datoteka = fopen(naziv_datoteke, "r");
    if (!datoteka) {

```

```

        printf("Greska pri otvaranju datoteke: %s\n", naziv_datoteke);
        return;
    }

    *n = 0;
    while (fscanf(datoteka, "%s\n%s\n%s\n", posete[*n].ime_ljubimca,
posete[*n].vrsta, posete[*n].datum_posete) == 3) {
        (*n)++;
    }

    fclose(datoteka);
}

void dodaj_posetu(POSETA p, POSETA posete[], int* n) {
    if (*n == 0) {
        posete[0] = p;
        (*n)++;
        return;
    }

    int i = *n - 1;
    while (i >= 0 && strcmp(p.datum_posete, posete[i].datum_posete) < 0)
    {
        posete[i + 1] = posete[i];
        i--;
    }

    posete[i + 1] = p;
    (*n)++;
}

void kreiraj_listu(PCVOR* glava, POSETA posete[], int n) {

    for (int i = 0; i < n; i++) {
        int postoji = 0;
        for (PCVOR trenutni = *glava; trenutni != NULL; trenutni =
trenutni->sledeci) {
            if (strcmp(posete[i].ime_ljubimca, trenutni->ime_ljubimca) ==
0 && strcmp(posete[i].vrsta, trenutni->vrsta) == 0) {
                postoji = 1;
                break;
            }
        }

        if (!postoji) {
            PCVOR noviCvor = malloc(sizeof(CVOR));
            strcpy(noviCvor->ime_ljubimca, posete[i].ime_ljubimca);
            strcpy(noviCvor->vrsta, posete[i].vrsta);
            noviCvor->sledeci = NULL;

            if (*glava == NULL) {
                *glava = noviCvor;
            }
            else {
                PCVOR trenutni = *glava;
                while (trenutni->sledeci != NULL) {
                    trenutni = trenutni->sledeci;
                }
                trenutni->sledeci = noviCvor;
            }
        }
    }
}

```

```

    }
}

void izbaci_iz_liste(PCVOR* glava, char ime[]) {
    if (*glava == NULL) {
        return;
    }

    PCVOR trenutni = *glava;
    PCVOR prethodni = NULL;
    //B, D, E
    while (trenutni != NULL) {
        if (strcmp(trenutni->ime_ljubimca, ime) == 0) {
            if (prethodni == NULL) {
                *glava = trenutni->sledeci; //glava = B
                free(trenutni);
                trenutni = *glava;
            }
            else {
                prethodni->sledeci = trenutni->sledeci;
                //prethodni.sledeci = D
                PCVOR za_brisanje = trenutni; //za brisanje = C
                trenutni = trenutni->sledeci; // trenutni = D
                free(za_brisanje); //brisemo C
            }
        }
        else {
            prethodni = trenutni; //prethodni = B
            trenutni = trenutni->sledeci; //trenutni = C
        }
    }
}

void kreiraj_matricu(POSETA posete[], int n, int statistika[3][12]) {
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 12; j++) {
            statistika[i][j] = 0;
        }
    }

    for (int i = 0; i < n; i++) {
        int dd, MM, yyyy;
        sscanf(posete[i].datum_posete, "%d.%d.%d.", &dd, &MM, &yyyy);

        int index_vrsta;
        if (strcmp(posete[i].vrsta, "Macka") == 0) {
            index_vrsta = 0;
        }
        else if (strcmp(posete[i].vrsta, "Pas") == 0) {
            index_vrsta = 1;
        }
        else {
            index_vrsta = 2;
        }

        statistika[index_vrsta][MM - 1]++;
    }
}

```

```

int najposeceniji_mesec(int statistika[3][12]) {
    int max_poseta = 0;
    int mesec = -1;

    for (int j = 0; j < 12; j++) {
        int broj_poseta = statistika[0][j] + statistika[1][j] +
statistika[2][j];
        if (broj_poseta > max_poseta) {
            max_poseta = broj_poseta;
            mesec = j + 1;
        }
    }

    return mesec;
}

//POMOCNE FUNKCIJE

void prikazi_niz(POSETA posete[], int n) {
    for (int i = 0; i < n; i++) {
        printf("Ime ljubimca: %s\n", posete[i].ime_ljubimca);
        printf("Vrsta ljubimca: %s\n", posete[i].vrsta);
        printf("Datum posete: %s\n", posete[i].datum_posete);
        printf("-----\n");
    }
}

void prikazi_listu(PCVOR glava) {
    PCVOR trenutni = glava;
    while (trenutni != NULL) {
        printf("Ime ljubimca: %s\n", trenutni->ime_ljubimca);
        printf("Vrsta ljubimca: %s\n", trenutni->vrsta);
        printf("-----\n");
        trenutni = trenutni->sledeci;
    }
}

void prikaz_matrice(int statistika[3][12]) {
    char* meseci[] = { "Jan", "Feb", "Mar", "Apr", "Maj", "Jun", "Jul",
"Avg", "Sep", "Okt", "Nov", "Dec" };
    char* vrste[] = { "Macka", "Pas", "Ostale" };

    printf("%7s", " ");
    for (int i = 0; i < 12; i++) {
        printf("| %s ", meseci[i]);
    }
    printf("| \n");

    printf("\t-----
-----\n");

    for (int i = 0; i < 3; i++) {
        printf("%7s", vrste[i]);
        for (int j = 0; j < 12; j++) {
            printf("| %3d ", statistika[i][j]);
        }
        printf("| \n");
        printf("\t-----
-----\n");
    }
}

```

```

}

int main() {

    // 1. Provera datuma
    int rezultat1 = format_datuma("25.06.2023.");
    int rezultat2 = format_datuma("30.02.2022.");
    int rezultat3 = format_datuma("12.13.2021.");

    printf("Datum %s je %s\n", "25.06.2023.", rezultat1 ? "ispravan" :
"neispravan");
    printf("Datum %s je %s\n", "30.02.2022.", rezultat2 ? "ispravan" :
"neispravan");
    printf("Datum %s je %s\n", "12.13.2021.", rezultat3 ? "ispravan" :
"neispravan");

    // 2. Ucitavanje poseta
    POSETA posete[100];
    int n = 0;
    ucitaj_posete("posete.txt", posete, &n);
    printf("\nPosete u nizu su:\n");
    prikazi_niz(posete, n);

    // 3. Dodavanje poseta u niz
    POSETA nova_poseta;
    strcpy(nova_poseta.ime_ljubimca, "Bela");
    strcpy(nova_poseta.vrsta, "Pas");
    strcpy(nova_poseta.datum_posete, "26.6.2023.");
    dodaj_posetu(nova_poseta, posete, &n);
    printf("\nPosete nakon dodavanja nove posete su:\n");
    prikazi_niz(posete, n);

    // 4. Kreiranje liste
    PCVOR glava = NULL;
    kreiraj_listu(&glava, posete, n);
    printf("\nPosete u listi su:\n");
    prikazi_listu(glava);

    // 5. Izbacivanje ljubimaca iz liste
    izbaci_iz_liste(&glava, "Garfield");
    printf("\nPosete nakon izbacivanja su:\n");
    prikazi_listu(glava);

    // 6. Kreiranje matrice
    int statistika[3][12];
    POSETA posete2[50] = {
        {"Bela", "Pas", "01.01.2023."},
        {"Mimi", "Macka", "15.02.2023."},
        {"Rex", "Pas", "10.03.2023."},
        {"Luna", "Macka", "20.04.2023."},
        {"Max", "Pas", "05.05.2023."},
        {"Dzeki", "Pas", "07.06.2023."},
        {"Kica", "Ptica", "07.06.2023."},
        {"Maza", "Zec", "25.07.2023."},
        {"Deki", "Ribica", "18.08.2023."},
        {"Tara", "Pauk", "11.09.2023."},
        {"Rob", "Zmija", "02.10.2023."},
        {"Bruno", "Pas", "19.12.2023."},
        {"Lola", "Macka", "03.01.2023."},
        {"Caki", "Pas", "12.02.2023."},
    };
}

```



```

        {"Riki", "Pas", "26.03.2023."},
        {"Laki", "Macka", "08.03.2023."},
        {"Dora", "Macka", "21.05.2023."},
        {"Nora", "Papagaj", "12.02.2023."},
        {"Maki", "Krava", "26.03.2023."},
        {"Laki", "Konj", "08.04.2023."},
        {"Munja", "Vrabac", "22.09.2023."},
        {"Lola", "Pas", "04.10.2023."},
        {"Maza", "Nosorog", "06.11.2023."},
        {"Max", "Pas", "15.12.2023."},
        {"Rex", "Pas", "26.01.2023."},
    };
    int n2 = 25;
    kreiraj_matricu(posete2, n2, statistika);
    printf("\n\n");
    prikaz_matrice(statistika);

    // 7. Najposeceniji mesec
    int najposeceniji = najposeceniji_mesec(statistika);
    printf("\nNajposeceniji mesec: %d\n", najposeceniji);

    return 0;
}

```



## ПРИНЦИПИ ПРОГРАМИРАЊА (ПРОГРАМИРАЊЕ I)

### Практични део испита

Име и презиме: \_

Бр. индекса:

Поени: \_

#### Обавештења:

- Испит траје 120 минута.
- На локалном диску С направити фолдер под називом JUL\_T2G3.
- НАЗИВ ФАЈЛА **ИЗВОРНОГ КОДА** МОРА БИТИ У СЛЕДЕЋЕМ ФОРМАТУ име\_презиме\_годинауписа\_бројиндекса (пример: Pera\_Peric\_2021\_0001.c). Сви задаци се раде у једном пројекту, у једној .с датотеци.
- За сваки задатак где је дат прототип функције мора да се имплементира функција која одговара прототипу (ИСТИ назив, листа параметара и повратна вредност), исто важи и за дате типове.
- Могу се дописивати додатне помоћне функције и типови.

#### Задатак:

Дата је следећа структура која представља песму:

```
typedef struct pesma{
    int id_albuma;
    int id_pesme;
    char naziv[50];
    char trajanje[6]; //trajanje pesme je u
    formatu mm:ss
} PESMA;
```

```
typedef struct cvor CVOR;
typedef struct cvor * PCVOR;
struct cvor{
    int sifra_albuma;
    int sifra_pesme;
    int trajanje; // trajanje u sekundama
    PCVOR sledeci;
};
```

Имплементирати следеће функције:

#### ▪ void unesi\_pesme(PCVOR \* glava)

15п.

Функција омогућава кориснику унос песама у листу, све док корисник то жели. Након уноса података о песми, потребно је питати корисника да ли жели да понови унос. Уколико жели, корисник треба да унесе карактер d, уколико не жели треба да унесе карактер n, након чега се прекида даљи унос.

Пример рада са конзолом:

```
Unesite id albuma:
0
Unesite id pesme:
123
Unesite naziv pesme:
Let it be
Unesite trajanje:
03:50
Da li zelite da unesete jos pesama? (d/n)
d
Unesite id albuma:
```

```
0
Unesite id pesme:
159
Unesite naziv pesme:
Yesterday
Unesite trajanje:
02:03
Da li zelite da unesete jos pesama? (d/n)
n
```

#### ▪ int vreme\_trajanja(char vreme[])

15п.

Функција као улазни параметар прихвата стринг који представља трајање песме у минутима и секундама, у формату mm:ss (на пример 03:30). Функција треба да врати трајање песме у секундама. Уколико је број минута већи од 60 и/или број секунди већи од 59, функција треба да врати -1.

#### ▪ void ubaci\_u\_sortiranu(PESMA p, PCVOR \* glava)

20п.

Функција додаје податке о песми у листу сортирану по дужини трајања песме. Након додавања новог чвора листа мора да буде сортирана у неоппадајућем редоследу по дужини трајању песме.

(За 10 од 20п. улазна листа је сортирана по шифри песме, и потребно је додати песму у листу, тако да она остане сортирана по шифри).

#### ▪ void kreiraj\_album(int id\_albuma, PCVOR glava, PESMA pesme[], int \* n);

15п.

Од листе је потребно направити низ песама који садржи све песме са албума са прослеђеном шифром. Песме треба додавати на крај низа.



▪ **void ogranicit\_trajanje(PESMA album[], int \* n)**

**15п.**

Албум се састоји од песама. Најдуже трајање албума може бити 1800 секунди. Потребно је имплементирати функцију која из низа избацује песме из албума све док албум не испуњава услов. Песме из низа се избацују тако што се прво избаци песма са најмањом вредношћу, уколико је трајање албума и даље дужа од 1800 секунди, опет се избацује песма са најмањом вредношћу, и тако све док трајање албума не буде мање или једнако 1800 сек.

▪ **void ucitaj\_slusanja(char\* naziv\_datoteke, int id\_albuma, int stats[][10], int \* n)**

**20п.**

У датотеци се налазе подаци о преслушавањима песама албума од стране разних корисника. Тескуална датотека је форматирана тако што се у првом реду налази информација о шифри албума и броју узорка, а затим се у сваком реду налазе подаци о преслушавањима. Претпоставити да сваки албум има тачно 10 песама. У матрицу статистика потребно је учитати статистику о преслушавањима само за албум са прослеђеном шифром. Параметар n представља тренутни број редова у матрици.

Формат датотеке:

```
шифра - албума, узорак
преслушавање-песме1, преслушавање-песме2, ..., преслушавање-песме10
преслушавање-песме1, преслушавање-песме2, ..., преслушавање-песме10
преслушавање-песме1, преслушавање-песме2, ..., преслушавање-песме10
```

Пример датотеке:

```
5, 2
9, 5, 2, 4, 7, 9, 4, 1, 0, 8
12, 5, 42, 7, 0, 0, 6, 11, 0, 8
1, 3
5, 26, 23, 1, 7, 0, 34, 2, 9, 0
6, 4, 2, 7, 9, 32, 78, 23, 43, 12
25, 13, 56, 54, 23, 65, 12, 1, 6, 23
```

На пример, уколико је потребно учитати статистику за албум са шифром 1, онда ће матрица бити:

5	26	23	1	7	0	34	2	9	0
6	4	2	7	9	32	78	23	43	12
25	13	56	54	23	65	12	1	6	23

Док n треба да има вредност 3.

▪ **int najslusanija\_pesma(int stats[][10], int n)**

**20п.**

Функција враћа индекс колоне која представља најслушанију песму од стране највише корисника. Односно потребно је пронаћи песму која има највећи фактор популарности који се рачуна по следећој формули: Фактор популарности = укупан број преслушавања \* број јединствених корисника који су преслушали песму (број преслушавања је већи од 0).

Инд.	0	1	2	3	4	5	6	7	8	9
	5	6	0	1	7	0	3	12	9	0
	6	4	95	7	9	2	7	11	4	1
	5	3	0	4	3	5	1	15	6	2
Инд. поп.	16*3=48	13*3=39	95*1=95	12*3=36	19*3=57	7*2=14	11*3=33	38*3=114	19*3=57	3*2=6

Функција треба да врати вредност 7, јер се у тој колони налази песма са највећим индексом популарности. (За 10 од 20 поена, вратити индекс колоне која садржи највећу суму преслушавања)

```

#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <ctype.h>
#include <math.h>

//JUL 2023 - GRUPA 3

typedef struct pesma {
    int id_albuma;
    int id_pesme;
    char naziv[50];
    char trajanje[6]; // trajanje pesme je u formatu mm:ss
} PESMA;

typedef struct cvor CVOR;

typedef struct cvor* PCVOR;

struct cvor {
    int sifra_albuma;
    int sifra_pesme;
    char naziv[50];
    int trajanje; // trajanje u sekundama
    PCVOR sledeci;
};

//GLAVNE FUNKCIJE

void ubaci_u_sortiranu(PESMA p, PCVOR* glava);

void unesi_pesme(PCVOR* glava) {
    char unos;
    do {
        PESMA nova_pesma;
        printf("Unesite id albuma:\n");
        scanf("%d", &nova_pesma.id_albuma);
        getchar();

        printf("Unesite id pesme:\n");
        scanf("%d", &nova_pesma.id_pesme);
        getchar();

        printf("Unesite naziv pesme:\n");
        fgets(nova_pesma.naziv, sizeof(nova_pesma.naziv), stdin);
        int len = strlen(nova_pesma.naziv);
        if (nova_pesma.naziv[len - 1] == '\n') {
            nova_pesma.naziv[len - 1] = '\0';
        }

        printf("Unesite trajanje u formatu mm:ss:\n");
        scanf("%s", &nova_pesma.trajanje);
        getchar();

        ubaci_u_sortiranu(nova_pesma, glava);

        printf("Da li zelite da unesete jos pesama? (d/n)\n");
        scanf("%c", &unos);
    } while (unos == 'd');
}

```

```

}

int vreme_trajanja(char vreme[]) {
    int minute, sekunde;
    if (sscanf(vreme, "%d:%d", &minute, &sekunde) != 2) {
        return -1; // Neispravan format vremena
    }

    if (minute > 60 || sekunde > 59) {
        return -1; // Neispravno vreme (vise od 60 minuta ili 59 sekundi)
    }

    return minute * 60 + sekunde;
}

void ubaci_u_sortiranu(PESMA p, PCVOR* glava) {
    PCVOR noviCvor = malloc(sizeof(CVOR));
    if (noviCvor == NULL) {
        printf("Greska pri alokaciji memorije!\n");
        return;
    }

    noviCvor->sifra_albuma = p.id_albuma;
    noviCvor->sifra_pesme = p.id_pesme;
    noviCvor->trajanje = vreme_trajanja(p.trajanje);
    strcpy(noviCvor->naziv, p.naziv);
    noviCvor->sledeci = NULL;

    if (*glava == NULL || noviCvor->trajanje < (*glava)->trajanje) {
        noviCvor->sledeci = *glava;
        *glava = noviCvor;
        return;
    }

    //3 5 9 15 18 -> 10

    PCVOR trenutni = *glava;
    while (trenutni->sledeci != NULL && trenutni->sledeci->trajanje <=
noviCvor->trajanje) {
        trenutni = trenutni->sledeci;
    }

    noviCvor->sledeci = trenutni->sledeci; // 10 -> 15
    trenutni->sledeci = noviCvor; //9 -> 10
}

void kreiraj_album(int id_albuma, PCVOR glava, PESMA pesme[], int* n) {
    *n = 0;
    PCVOR trenutni = glava;

    while (trenutni != NULL && *n < 10) {
        if (trenutni->sifra_albuma == id_albuma) {
            pesme[*n].id_albuma = trenutni->sifra_albuma;
            pesme[*n].id_pesme = trenutni->sifra_pesme;
            strcpy(pesme[*n].naziv, trenutni->naziv);
            char trajanje_str[6];
            sprintf(trajanje_str, "%d", trenutni->trajanje);
            strcpy(pesme[*n].trajanje, trajanje_str);
            (*n)++;
        }
    }
}

```

```

        trenutni = trenutni->sledeci;
    }
}

void ogranic_i_trajanje(PESMA album[], int* n) {
    int ukupno_trajanje = 0;

    for (int i = 0; i < *n; i++) {
        ukupno_trajanje += atoi(album[i].trajanje);
    }

    while (ukupno_trajanje > 1800 && *n > 0) {
        ukupno_trajanje -= atoi(album[0].trajanje);
        for (int i = 0; i < *n - 1; i++) {
            album[i] = album[i + 1];
        }
        (*n)--;
    }
}

void ucitaj_slusanja(char* naziv_datoteke, int id_albuma, int
stats[][10], int* n) {
    FILE* datoteka = fopen(naziv_datoteke, "r");
    if (datoteka == NULL) {
        printf("Greska pri otvaranju datoteke!\n");
        return;
    }

    int sifra_albuma, uzorak;
    while (fscanf(datoteka, "%d,%d\n", &sifra_albuma, &uzorak) == 2) {
        if (sifra_albuma == id_albuma) {
            for (int uzorak_br = 0; uzorak_br < uzorak; uzorak_br++) {
                for (int i = 0; i < 10; i++) {
                    if (i != 9) {
                        fscanf(datoteka, "%d,", &stats[*n][i]);
                    }
                    else {
                        fscanf(datoteka, "%d\n", &stats[*n][i]);
                    }
                }
                (*n)++;
            }
            break;
        }
        else {
            char temp[100];
            for (int i = 0; i < uzorak; i++) {
                fgets(temp, 100, datoteka);
            }
        }
    }

    fclose(datoteka);
}

int najslusanija_pesma(int stats[][10], int n) {
    int max_popularnost = -1;
    int indeks_max_popularnosti = -1;

    for (int j = 0; j < 10; j++) {

```

```

        int broj_preslusavanja = 0;
        int broj_korisnika = 0;

        for (int i = 0; i < n; i++) {
            if (stats[i][j] > 0) {
                broj_preslusavanja += stats[i][j];
                broj_korisnika++;
            }
        }

        int popularnost = broj_preslusavanja * broj_korisnika;
        if (popularnost > max_popularnost) {
            max_popularnost = popularnost;
            indeks_max_popularnosti = j;
        }
    }

    return indeks_max_popularnosti;
}

//POMOCNE FUNKCIJE

void ispisi_album(PESMA pesme[], int n) {
    for (int i = 0; i < n; i++) {
        printf("ID pesme: %d, Trajanje: %s Naziv: %s\n",
            pesme[i].id_pesme, pesme[i].trajanje, pesme[i].naziv);
    }
}

void ispis_statistike(int stats[][10], int n) {
    printf("\nStatistika preslusavanja:\n");
    for (int i = 0; i < n; i++) {
        printf("Preslusavanje pesme %d: ", i + 1);
        for (int j = 0; j < 10; j++) {
            printf("%2d ", stats[i][j]);
        }
        printf("\n");
    }
}

int main() {

    //1. Unos pesama
    PCVOR glava = NULL;
    unesi_pesme(&glava);
    PESMA data[] = {
        { 0, 101, "You're Going to Lose That Girl", "02:19" },
        { 0, 102, "Ticket to Ride", "03:10" },
        { 0, 103, "It's Only Love", "01:53" },
        { 0, 104, "Help!", "02:18" },
        { 0, 105, "Tell Me What You See", "02:37" },
        { 0, 106, "Another Girl", "02:05" },
        { 0, 107, "Act Naturally", "02:32" },
        { 0, 108, "I Need You", "02:28" },
        { 0, 109, "You Like Me Too Much", "02:36" },
    };
    int brojPesama = sizeof(data) / sizeof(data[0]);
    for (int i = 0; i < brojPesama; i++) {
        ubaci_u_sortiranu(data[i], &glava);
    }
}

```

```

// 2. Kreiranje albuma
PESMA pesme[10];
int n;
kreiraj_album(0, glava, pesme, &n);
printf("Album je:\n");
ispisi_album(pesme, n);

// 3. Trajanje albuma
ogranici_trajanje(pesme, &n);
printf("\nAlbum nakon ogranichenja trajanja je:\n");
ispisi_album(pesme, n);

// 4. Ucitavanje statistike
int stats[10][10];
int m = 0;
ucitaj_slusanje("slusanje.txt", 1, stats, &m);
ispis_statistike(stats, m);

// 5. Najslusanija pesma
int najslusanija = najslusanija_pesma(stats, m);
printf("\nNajslusanija pesma (indeks kolone): %d\n", najslusanija);

return 0;
}

```





ПРОГРАМИРАЊЕ I - Практични део испита

Име и презиме:

Бр. индекса:

Поени:

**Обавештења:**

- Испит траје 120 минута.
- На локалном диску С направити фолдер под називом SEPT\_T2G1.
- НАЗИВ ФАЈЛА **ИЗВОРНОГ КОДА** МОРА БИТИ У СЛЕДЕЋЕМ ФОРМАТУ име\_презиме\_годинауписа\_бројиндекса (пример: Pera\_Peric\_2021\_0001.c). Сви задаци се раде у једном пројекту, у једној .c датотеци.
- За сваки задатак где је дат прототип функције мора да се имплементира функција која одговара прототипу (ИСТИ назив, листа параметара и повратна вредност), исто важи и за дате типове.
- Могу се дописивати додатне помоћне функције и типови.

**Задатак:**

Дата је следећа структура која представља прогнозу за одређени дан и месец:

```
typedef struct prognoza{
    int min_temp;
    int max_temp;
    int padavine; //kolicine padavina
    int mesec;
    int dan;
} PROGNOZA;

typedef struct cvor CVOR;
typedef struct cvor *PCVOR;
struct cvor{
    PROGNOZA prognoza;
    PCVOR sledeci;
};
```

Потребно је имплементирати следеће функције:

- **void ucitaj\_padavine(char \* naziv\_datoteke, int padavine[], int \* n) 10п.**  
Функција учитава податке о падавинама из текстуалне датотеке у прослеђени низ. Број редова у датотеци није унапред познат.  
Пример датотеке:  
0  
10  
11  
0  
2  
0  
1  
0
- **int max\_kisa(int padavine[], int n) 15п.**  
Параметар *padavine* представља низ целих бројева који представљају количину падавина у литрима за сваки дан. Функција треба да израчуна и врати укупну количину падавина у литрима која је пала током најдужега узастопног периода са кишом, односно треба одредити највећу количину кише која је пала у узастопним данима.  
За низ:  
0 0 1 0 2 5 0 1 3 1 0 2  
Функција треба да врати 7, јер је подниз са највећом сумом 2 5.
- **int susni\_dani(int padavine[], int n) 10п.**  
Функција треба да врати број сушних дана, дана без кише када је количина падавина била једнака 0).  
За низ:  
0 0 1 0 2 5 0 1 3 1 0 2  
Функција треба да врати 5.



▪ **void dodaj\_temperaturu(int temperature[][24], int \* n) 15п.**

Имплементирати функцију која у матрицу додаје податке о температури за један дан. Матрица садржи тачно 24 колоне које представљају температуру за сваки сат у дану, док један ред представља податке о температурама за један дан. Индекс 0 представља сат 0, индекс 1 представља 1 сат итд.

Функција треба да омогући додавање тачно једног новог реда у матрицу. Прослеђена матрица може бити празна или може имати већ унете редове.

Unesite temperature po satima (potrebno je uneti 24 vrednosti):

12

12

13

13

—

12

Kraj unosa!

▪ **int najhladniji\_sat(int temperature[][24], int n) 20п.**

Функција треба да врати који сат је просеку најхладнији, односно у просеку има најнижу температуру.

15	14	14	13	13	13	13	14	15	17	18	19	20	20	21	22	22	22	22	21	19	17	<b>17</b>	16
17	18	18	18	19	19	19	19	20	21	22	22	23	23	24	25	25	26	27	26	25	22	<b>22</b>	22
-10	-8	-8	-8	-8	-8	-7	-7	-7	-5	-5	-5	-5	-5	-4	-5	-5	-5	-7	-7	-7	-8	<b>-30</b>	-8
-10	-8	-8	-8	-8	-8	-7	-7	-7	-5	-5	-5	-5	-5	-4	-5	-5	-5	-7	-7	-7	-8	<b>-30</b>	-8
-10	-8	-8	-8	-8	-8	-7	-7	-7	-5	-5	-5	-5	-5	-4	-5	-5	-5	-7	-7	-7	-8	<b>-30</b>	-8
-10	-8	-8	-7	-7	-6	-5	-3	-2	0	1	1	1	0	0	1	1	1	1	2	2	2	<b>2</b>	3
-5	0	0	0	0	0	1	2	2	4	4	5	5	5	6	6	6	6	5	5	4	3	<b>-8</b>	3

Функција треба да врати 22 јер је то индекс колоне, односно сат која има најмању просечну вредност.

▪ **void dodaj\_u\_listu(PROGNOZA prognoza, PCVOR \* glava) 10п.**

Функција додаје нови чвор на крај листе.

▪ **int prover\_i\_padavine(int dan\_od, int mesec\_od, int dan\_do, int mesec\_do, PCVOR glava) 20п.**

Функција треба да врати да ли ће у прослеђеном временском интервалу падати киша. Функција треба да врати 1 уколико ће бити падавина у прослеђеном периоду, односно 0 уколико неће бити падавина.

Уколико листа има следеће податке:

Dan: 15, Mesec: 6, Min temp: 22°C, Max temp: 32°C, Padavine: 3

Dan: 16, Mesec: 6, Min temp: 21°C, Max temp: 30°C, Padavine: 6

Dan: 1, Mesec: 8, Min temp: 15°C, Max temp: 25°C, Padavine: 5

Dan: 2, Mesec: 8, Min temp: 18°C, Max temp: 27°C, Padavine: 0

Dan: 3, Mesec: 8, Min temp: 20°C, Max temp: 30°C, Padavine: 10

Dan: 10, Mesec: 7, Min temp: 23°C, Max temp: 33°C, Padavine: 2

Dan: 11, Mesec: 7, Min temp: 25°C, Max temp: 35°C, Padavine: 0

За унети датум од 1.8. и датум до 3.8. функција треба да врати 1 (5+0+10 је веће од 0).

```

#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <ctype.h>
#include <math.h>

//SEPTEMBAR 2023

typedef struct prognoza {
    int min_temp;
    int max_temp;
    int padavine;
    int mesec;
    int dan;
} PROGNOZA;

typedef struct cvor CVOR;

typedef struct cvor* PCVOR;

typedef struct cvor {
    PROGNOZA prognoza;
    PCVOR sledeci;
};

void ucitaj_padavine(char* naziv_datoteke, int padavine[], int* n) {
    FILE* datoteka = fopen(naziv_datoteke, "r");
    if (datoteka == NULL) {
        printf("Greska prilikom otvaranja datoteke.\n");
        return;
    }

    while (fscanf(datoteka, "%d", &padavine[*n]) == 1) {
        (*n)++;
    }

    fclose(datoteka);
}

int max_kisa(int padavine[], int n) {
    int max_suma = 0;
    int trenutna_suma = 0;

    for (int i = 0; i < n; i++) {
        if (padavine[i] > 0) {
            trenutna_suma += padavine[i];
            if (trenutna_suma > max_suma) {
                max_suma = trenutna_suma;
            }
        }
        else {
            trenutna_suma = 0;
        }
    }
    return max_suma;
}

int susni_dani(int padavine[], int n) {
    int broj_susnih_dana = 0;

```

```

        for (int i = 0; i < n; i++) {
            if (padavine[i] == 0) {
                broj_susnih_dana++;
            }
        }
        return broj_susnih_dana;
    }

void dodaj_temperaturu(int temperature[][24], int* n) {
    printf("Unesite temperature po satima (potrebno je uneti 24
vrednosti):\n");
    for (int i = 0; i < 24; i++) {
        scanf("%d", &temperature[*n][i]);
    }
    printf("Kraj unosa!\n");
    (*n)++;
}

int najhladniji_sat(int temperature[][24], int n) {
    double prosek_temp[24] = { 0 };
    for (int j = 0; j < 24; j++) {
        for (int i = 0; i < n; i++) {
            prosek_temp[j] += temperature[i][j];
        }

        for (int i = 0; i < 24; i++) {
            prosek_temp[i] /= n;
        }
        int najhladniji_sat = 0;
        for (int i = 1; i < 24; i++) {
            if (prosek_temp[i] < prosek_temp[najhladniji_sat]) {
                najhladniji_sat = i;
            }
        }
        return najhladniji_sat;
    }
}

void dodaj_u_listu(PROGNOZA prognoza, PCVOR* glava) {
    PCVOR noviCvor = malloc(sizeof(CVOR));
    if (noviCvor == NULL) {
        printf("Greska pri alokaciji memorije!\n");
        return;
    }
    noviCvor->prognoza = prognoza;
    noviCvor->sledeci = NULL;
    if (*glava == NULL) {
        *glava = noviCvor;
    }
    else {
        PCVOR trenutni = *glava;
        while (trenutni->sledeci) {
            trenutni = trenutni->sledeci;
        }
        trenutni->sledeci = noviCvor;
    }
}

int prover_i_padavine(int dan_od, int mesec_od, int dan_do, int mesec_do,
PCVOR glava) {

```

```

        while (glava) {
            if ((glava->prognoza.mesec > mesec_od || (glava->prognoza.mesec
== mesec_od && glava->prognoza.dan >= dan_od)) &&
                (glava->prognoza.mesec < mesec_do || (glava->prognoza.mesec
== mesec_do && glava->prognoza.dan <= dan_do)) &&
                    glava->prognoza.padavine > 0) {
                return 1;
            }
            glava = glava->sledeci;
        }
        return 0;
    }

int main() {

    //01. Ucitavanje padavina
    int padavine[100];
    int n = 0;
    ucitaj_padavine("padavine.txt", padavine, &n);

    //02. Provera maksimalne kolicine padavina
    printf("Maksimalna kolicina padavina u uzastopnim danima: %d\n",
max_kisa(padavine, n));

    //03. Provera broja susnih dana
    printf("Broj susnih dana: %d\n", susni_dani(padavine, n));

    //04. Unos temperature
    int temperature[365][24] = {
        {15, 14, 14, 13, 13, 13, 13, 14, 15, 17, 18, 19, 20, 20, 21, 22,
22, 22, 22, 21, 19, 17, 17, 16},
        {17, 18, 18, 18, 19, 19, 19, 19, 20, 21, 22, 22, 23, 23, 24, 25,
25, 26, 27, 26, 25, 22, 22, 22},
        {-10, -8, -8, -8, -8, -8, -7, -7, -7, -5, -5, -5, -5, -5, -4, -5,
-5, -5, -7, -7, -7, -8, -30, -8},
        {-10, -8, -8, -8, -8, -8, -7, -7, -7, -5, -5, -5, -5, -5, -4, -5,
-5, -5, -7, -7, -7, -8, -30, -8},
        {-10, -8, -8, -8, -8, -8, -7, -7, -7, -5, -5, -5, -5, -5, -4, -5,
-5, -5, -7, -7, -7, -8, -8, -8}
    };
    int broj_dana_temp = 5;
    //dodaj_temperaturu(temperature, &broj_dana_temp);
    printf("Najhladniji sat je: %d\n", najhladniji_sat(temperature,
broj_dana_temp));

    //05. Dodavanje prognoza u listu
    PCVOR glava = NULL;
    PROGNOZA prognoze[] = {
        {22, 32, 3, 6, 15},
        {21, 30, 6, 6, 16},
        {15, 25, 5, 8, 1},
        {18, 27, 0, 8, 2},
        {20, 30, 10, 8, 3},
        {23, 33, 2, 7, 10},
        {25, 35, 0, 7, 11}
    };
    int brojPrognoza = sizeof(prognoze) / sizeof(prognoze[0]);
    for (int i = 0; i < brojPrognoza; i++) {
        dodaj_u_listu(prognoze[i], &glava);
    }
}

```

```
    //06. Provera padavina u određenom intervalu
    printf("Da li ce biti padavina u periodu 1.8. - 3.8.: %d\n",
proveri_padavine(1, 8, 3, 8, glava));

    return 0;
}
```



## ПРИНЦИПИ ПРОГРАМИРАЊА (ПРОГРАМИРАЊЕ I)

### Практични део испита

Име и презиме: \_

Бр. индекса:

Поени: \_

#### Обавештења:

- Испит траје 120 минута.
- На локалном диску С направити фолдер под називом ОКТ\_T1G1.
- НАЗИВ ФАЈЛА **ИЗВОРНОГ КОДА** МОРА БИТИ У СЛЕДЕЋЕМ ФОРМАТУ име\_презиме\_годинауписа\_бројиндекса (пример: Pera\_Peric\_2021\_0001.c). Сви задаци се раде у једном пројекту, у једној .c датотеци.
- За сваки задатак где је дат прототип функције мора да се имплементира функција која одговара прототипу (ИСТИ назив, листа параметара и повратна вредност), исто важи и за дате типове.
- Могу се дописивати додатне помоћне функције и типови.

#### Задатак:

Потребно је имплементирати следеће функције:

- **void mat\_u\_niz(int mat[][20], int n, int niz[], int \*nx)** **15п.**  
Матрица *mat* представља квадратну матрицу максималне димензије 20. Параметар *n* представља димензију прослеђене матрице. Функција треба да пребаци све бројеве који се налазе испод главне дијагонале, тако што се прво пребацују бројеви који се налазе у првој колони испод главне дијагонале, затим из друге колоне и тако све до последње. Претпоставити да је улазни низ празан. Параметар *nx* представља број чланова низа.

Пример:

За дату матрицу димензије 5:

```
1 2 3 4 5
6 7 8 9 10
11 12 13 14 15
16 17 18 19 20
21 22 23 24 25
```

По извршавању функције низ би требало да садржи следеће елементе

0	1	2	3	4	5	6	7	8	9
6	11	16	21	12	17	22	18	23	24

(Напомена: Уколико функција пребацује бројеве ред по ред, уместо колону по колону, онда је могуће остварити 8 од 15 поена)

- **void rotiraj(int niz[], int n, int x, char smer)** **20п.**  
Функција треба да ротира низ за *x* места у лево или у десно. Уколико је за *smer* прослеђен карактер *l* онда низ треба да се помера у лево, а уколико је прослеђен карактер *d* онда треба да се низ помери у десно. Целобројна вредност *n* представља број чланова низа.

Уколико је задат смер *l*, вредност параметра *x* је 3, а низ је:

3	8	0	5	2	6	1
---	---	---	---	---	---	---

По извршавању функције низ би требало да изгледа:

5	2	6	1	3	8	0
---	---	---	---	---	---	---

Уколико је задат смер *d*, вредност параметра *x* је 4, а низ је:

7	5	1	6	9	4	3
---	---	---	---	---	---	---

По извршавању функције низ би требало да изгледа:

6	9	4	3	7	5	1
---	---	---	---	---	---	---

(Напомена: уколико функција без обзира на прослеђен смер увек помера низ у лево или у десно остварује се 10 од 20п.)



Дати су следећи типови:

```
typedef struct cvor CVOR;  
typedef struct cvor * PCVOR;  
struct cvor{  
    int broj;  
    PCVOR sledeci;  
};
```

- **void dodaj(PCVOR \* glava, int broj)** **10п.**  
Функција треба да омогући додавање новог чвора на крај листе.
- **int frekvencija(PCVOR glava, int broj)** **10п.**  
Функција враћа број понављања броја *broj* у прослеђеној листи.
- **void presek(PCVOR lista1, PCVOR lista2, PCVOR \* lista3)** **20п.**  
Функција креира листу *lista3* која представља пресек листе *lista1* и *lista2*. Пресек обухвата оне елементе који се налазе и у листи *lista1* и у листи *lista2*. Претпоставити да се у листама *lista1* и *lista2* бројеви не понављају.  
Пример:  
*lista1*->1->5->2->8->NULL  
*lista2*->3->1->6->8->9->NULL  
*lista3*->1->8->NULL
- **char sifruj(char slovo, int n)** **15п.**  
Функција као улазни параметар прихвата карактер *slovo* и целобројну вредност *n*. Уколико је карактер мало слово, функција треба да помери позицију тог слова у алфabetском реду за *n* места унапред, уколико се дође до краја алфабета онда је потребно кренути испочетка. Што значи да за прослеђено *n=3*, карактер *a* ће постати *d*, док ће *z* постати *c*. Функција враћа шифровано слово. Уколико функцији није прослеђено мало слово онда функција треба да врати малу црту (-).  
(Напомена: уколико функција само помера алфabetски ред за *n*, и не проверава да ли је крај алфабета или не, могуће је остварити 8 од 15п.)
- **void prebaci(char \* stara\_datoteka, char \* nova\_datoteka)** **10п.**  
Функција као улазне параметре прихвата назив старе и назив нове датотеке. У старој датотеци се налази текст. Функција треба да прочита текст из старе датотеке, шифрује га и упише у нову датотеку.  
(Напомена: Уколико функција само пребацује текст из старе у нову датотеку, онда је могуће остварити 6 од 10 поена)



```

#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include <math.h>

//OKTOBAR 2023 - GRUPA 1

void mat_u_niz(int mat[][20], int n, int niz[], int* nx) {
    int i, j, k = 0;
    for (j = 0; j < n; j++) {
        for (i = 0; i < n; i++) {
            if (i > j) {
                niz[k++] = mat[i][j];
            }
        }
    }
    *nx = k;
}

void rotiraj(int niz[], int n, int x, char smer) {
    int i, j;
    int temp[7];
    if (smer == 'l') {
        x = x % n;
        for (i = 0; i < n; i++) {
            temp[i] = niz[(i + x) % n];
        }
        for (i = 0; i < n; i++) {
            niz[i] = temp[i];
        }
    }
    else if (smer == 'd') {
        x = n - (x % n);
        for (i = 0; i < n; i++) {
            temp[i] = niz[(i + x) % n];
        }
        for (i = 0; i < n; i++) {
            niz[i] = temp[i];
        }
    }
}

typedef struct cvor CVOR;
typedef struct cvor* PCVOR;

struct cvor {
    int broj;
    PCVOR sledeci;
};

void dodaj(PCVOR* glava, int broj) {
    PCVOR novi = malloc(sizeof(CVOR));
    novi->broj = broj;
    novi->sledeci = NULL;

    if (*glava == NULL) {
        *glava = novi;
    }
    else {

```

```

        PCVOR trenutni = *glava;
        while (trenutni->sledeci != NULL) {
            trenutni = trenutni->sledeci;
        }
        trenutni->sledeci = novi;
    }
}

int frekvencija(PCVOR glava, int broj) {
    int brojac = 0;
    while (glava != NULL) {
        if (glava->broj == broj)
            brojac++;
        glava = glava->sledeci;
    }
    return brojac;
}

void presek(PCVOR lista1, PCVOR lista2, PCVOR* lista3) {
    while (lista1 != NULL) {
        if (frekvencija(lista2, lista1->broj) > 0 && frekvencija(*lista3,
lista1->broj) == 0)
            dodaj(lista3, lista1->broj);
        lista1 = lista1->sledeci;
    }
}

char sifruj(char slovo, int n) {
    if (slovo >= 'a' && slovo <= 'z') {
        return (char)((slovo - 'a' + n) % 26 + 'a');
    }
    else {
        return '-';
    }
}

void prebaci(char* stara_datoteka, char* nova_datoteka) {
    FILE* stara = fopen(stara_datoteka, "r");
    FILE* nova = fopen(nova_datoteka, "w");

    if (stara == NULL || nova == NULL) {
        printf("Greska prilikom otvaranja datoteka.\n");
        return;
    }

    char c;
    while ((c = fgetc(stara)) != EOF) {
        if (c >= 'a' && c <= 'z') {
            fputc(sifruj(c, 3), nova); // šifrujemo svaki karakter sa
n=3
        }
        else {
            fputc(c, nova);
        }
    }

    fclose(stara);
    fclose(nova);
}

```

```
//POMOCNE FUNKCIJE
```

```
void ispisNiza(int niz[], int n) {
```

```
    printf("Niz: ");
    for (int i = 0; i < n; i++) {
        printf("%d ", niz[i]);
    }
    printf("\n");
}
```

```
void ispisListe(PCVOR lista) {
```

```
    PCVOR trenutni = lista;
    while (trenutni) {
        printf("%d -> ", trenutni->broj);
        trenutni = trenutni->sledeci;
    }
    printf("NULL\n");
}
```

```
char desifruj(char slovo, int n) {
```

```
    if (slovo >= 'a' && slovo <= 'z') {
        return (char)((slovo - 'a' - n + 26) % 26) + 'a');
    }
    else {
        return slovo;
    }
}
```

```
void vrati(char* nova_datoteka, char* stara_datoteka) {
```

```
    FILE* nova = fopen(nova_datoteka, "r");
    FILE* stara = fopen(stara_datoteka, "w");

    if (nova == NULL || stara == NULL) {
        printf("Greska prilikom otvaranja datoteka.\n");
        return;
    }
}
```

```
    char c;
```

```
    while ((c = fgetc(nova)) != EOF) {
        if (c >= 'a' && c <= 'z') {
```

```
            fputc(desifruj(c, 3), stara);    // Dešifrujemo svaki karakter
```

```
sa n=3
```

```
        }
        else {
            fputc(c, stara);
        }
    }
}
```

```
fclose(nova);
```

```
fclose(stara);
```

```
}
```

```
int main() {
```

```
    //01. Upis matrice u niz
```

```
    int mat[5][20] = {
        {1, 2, 3, 4, 5},
```

```

        {6, 7, 8, 9, 10},
        {11, 12, 13, 14, 15},
        {16, 17, 18, 19, 20},
        {21, 22, 23, 24, 25},
    };
    int niz[20];
    int n = 0;
    mat_u_niz(mat, 5, niz, &n);
    ispisNiza(niz, n);

    //02.01. Rotacija niza - levo
    int niz1[] = { 3, 8, 0, 5, 2, 6, 1 };
    int n1 = 7;
    rotiraj(niz1, n1, 3, 'l');
    ispisNiza(niz1, n1);

    //02.02. Rotacija niza - desno
    int niz2[] = { 7, 5, 1, 6, 9, 4, 3 };
    int n2 = 7;
    rotiraj(niz2, n2, 4, 'd');
    ispisNiza(niz2, n2);

    //03.00. Definisanje promenljivih
    PCVOR lista1 = NULL, lista2 = NULL, lista3 = NULL;
    int vrednostiLista1[] = { 1, 5, 3, 8, 5, 12, 18};
    int vrednostiLista2[] = { 3, 1, 6, 8, 9, 11, 12};

    //03.01. Ucitavanje vrednosti u listu1
    int velicinaLista1 = sizeof(vrednostiLista1) /
sizeof(vrednostiLista1[0]);
    for (int i = 0; i < velicinaLista1; i++) {
        dodaj(&lista1, vrednostiLista1[i]);
    }

    //03.02. Ucitavanje vrednosti u listu2
    int velicinaLista2 = sizeof(vrednostiLista2) /
sizeof(vrednostiLista2[0]);
    for (int i = 0; i < velicinaLista2; i++) {
        dodaj(&lista2, vrednostiLista2[i]);
    }

    //03.03. Racunanje preseka i prikaz liste
    presek(lista1, lista2, &lista3);
    ispisListe(lista3);

    //04.01. Sifrovanje teksta
    printf("Sifrovano slovo: %c\n", sifruj('z', 3));
    prebaci("datoteka1.txt", "datoteka2.txt");

    //04.02. Desifrovanje teksta
    vrati("datoteka2.txt", "datoteka3.txt");

    return 0;
}

```



## ПРИНЦИПИ ПРОГРАМИРАЊА (ПРОГРАМИРАЊЕ I)

### Практични део испита

Име и презиме: \_

Бр. индекса:

Поени: \_

#### Обавештења:

- Испит траје 120 минута.
- На локалном диску С направити фолдер под називом ОКТ\_T2G1.
- НАЗИВ ФАЈЛА **ИЗВОРНОГ КОДА** МОРА БИТИ У СЛЕДЕЋЕМ ФОРМАТУ име\_презиме\_годинауписа\_бројиндекса (пример: Pera\_Peric\_2021\_0001.c). Сви задаци се раде у једном пројекту, у једној .c датотеци.
- За сваки задатак где је дат прототип функције мора да се имплементира функција која одговара прототипу (ИСТИ назив, листа параметара и повратна вредност), исто важи и за дате типове.
- Могу се дописивати додатне помоћне функције и типови.

#### Задатак:

Потребно је имплементирати следеће функције:

- **void nuliraj(int mat[][20], int n, int m)** 20п.  
Матрица *mat* представља матрицу максималних димензија 20x20, чији су елементи позитивни цели бројеви. Параметар *n* представља број редова матрице, док *m* представља број колона. Функција треба да пронађе нуле у матрици и да елементе у колони и у реду где се та нула налази замени такође нулама. Водити рачуна да замену реда и колоне нулама, треба урадити само за нуле које су биле у иницијалној матрици. У матрици може постојати и више од једне нуле, такође обратити пажњу и на случај када се две или више нула иницијално налазе у истом реду или истој колони.

Примери:

За дату матрицу димензије 5, која иницијално садржи једну 0:

Иницијална матрица:

```
1 2 3 4 5
6 7 8 9 10
11 0 13 14 15
16 17 18 19 20
21 22 23 24 25
```

Резултујућа матрица

```
1 0 3 4 5
6 0 8 9 10
0 0 0 0 0
16 0 18 19 20
21 0 23 24 25
```

За дату матрицу димензије 5, која иницијално садржи више од једне 0:

Иницијална матрица:

```
1 2 3 4 5
6 7 8 9 10
11 0 13 14 15
16 17 18 0 20
21 22 23 24 25
```

Резултујућа матрица:

```
1 0 3 0 5
6 0 8 0 10
0 0 0 0 0
0 0 0 0 0
21 0 23 0 25
```

За дату матрицу димензије 5, која иницијално садржи више од једне 0 у истом реду или колони:

Иницијална матрица:

```
1 2 3 4 5
6 7 8 9 10
11 0 13 14 0
16 17 18 19 20
21 22 23 24 25
```

Резултујућа матрица:

```
1 0 3 4 0
6 0 8 9 0
0 0 0 0 0
16 0 18 19 0
21 0 23 24 0
```

(Напомена: уколико функција ради само када матрица има једну 0 – 10п., уколико ради и када има 2 нуле 15п., уколико ради у сва три случаја 20п.)

- **void izbaci\_duplikate(int niz[], int \* n)** 20п.  
Улазни низ је низ целих бројева, који може садржати бројеве који се понављају. Функција треба да избаци сва појављивања бројева, осим његовог последњег појављивања.

Пример:



Уколико је дат низ: 5 8 6 4 1 2 5 8 5 4 1, и  $n$  11,

Након извршавања функције низ треба да буде: 6 2 8 5 4 1

(Напомена: Уколико функција избацује сва појављивања, осим првог – 10 од 20п.)

Дати су следећи типови:

```
typedef struct cvor CVOR;  
typedef struct cvor * PCVOR;  
struct cvor{  
    int broj;  
    PCVOR sledeci;  
};
```

- **void dodaj(PCVOR \* glava, int broj)** 10п.  
Функција треба да омогући додавање новог чвора на почетак листе.
- **void prikazi(PCVOR glava)** 10п.  
Функција приказује садржај листе тако што прво прикаже редни број, а затим и вредност тог чвора. На пример:  
1. 5  
2. 8  
3. 6  
4. 3  
5. 2
- **void unija(PCVOR lista1, PCVOR lista2, PCVOR \* lista3)** 20п.  
Функција креира листу *lista3* која представља унију листи *lista1* и *lista2*. Унија обухвата све елементе који се налазе у листи *lista1* и у листи *lista2*.  
Пример:  
*lista1*->1->5->2->8->NULL  
*lista2*->3->1->6->8->9->NULL  
*lista3*->1->5->2->8->3->6->9->NULL
- **void velika\_u\_mala(char rec[])** 10п.  
Функција као улазни параметар прихвата стринг *rec*. Функција треба сва велика слова у речи да претвори у мала.  
Пре извршавања: *vEliKA*  
После извршавања: *velika*
- **int postoji(char \* naziv\_datoteke, char \* rec)** 10п.  
Функција као улазне параметре прихвата назив датотеке и стринг. Функција треба да провери да ли реч постоји у датотеци са прослеђеним називом. Улазни параметар *rec*, увек представља тачно једну реч (без размака), док је садржај датотеке текст. Улазни параметар *rec* је реч чија су сва слова мала. Обратити пажњу да у текстуалној датотеци могу постојати и речи са почетним великим словом, те их прво треба претворити у речи са малим словима, како би се могла упоредити са прослеђеном речју. Уколико у датотеци постоји прослеђена реч, онда функција треба да врати 1, у супротном функција треба да врати 0.  
Пример садржаја датотеке:  
*Lorem IPSUM Dolor sit amet consectetur adipiscing elit Sed do EIUSMOD tempor incidunt ut labore et DOLOEE magna ALIQUA Varius morbi enim Nunc faucibus vitae purus faucibus ORNARE suspendisse sed*  
Уколико је прослеђена реч:  
*lorem* функција треба да врати 1  
*programiranje* функција треба да врати 0  
*vitae* функција треба да врати 1  
(Напомена: Уколико функција не проналази реч када садржи једно или више великих слова, онда се функција бодује са 5 од 10 поена)

```

#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include <math.h>

//OKTOBAR 2023 - GRUPA 2

void nuliraj(int mat[][20], int n, int m) {
    int i, j;
    int redovi[20] = { 0 }, kolone[20] = { 0 };
    for (i = 0; i < n; i++) {
        for (j = 0; j < m; j++) {
            if (mat[i][j] == 0) {
                redovi[i] = 1;
                kolone[j] = 1;
            }
        }
    }
    for (i = 0; i < n; i++) {
        for (j = 0; j < m; j++) {
            if (redovi[i] == 1 || kolone[j] == 1) {
                mat[i][j] = 0;
            }
        }
    }
}

void izbaci_duplikate(int niz[], int *n) {
    for (int i = *n - 1; i >= 0; i--) {
        for (int j = i - 1; j >= 0; j--) {
            if (niz[i] == niz[j]) {
                for (int k = j; k < *n - 1; k++) {
                    niz[k] = niz[k + 1];
                }
                //5 8 6 2 5 8 5 4 1
                //          j i
                (*n)--;
                i--;
            }
        }
    }
}

typedef struct cvor CVOR;

typedef struct cvor* PCVOR;

struct cvor {
    int broj;
    PCVOR sledeci;
};

void dodaj(PCVOR* glava, int broj) {
    PCVOR noviCvor = malloc(sizeof(CVOR));
    noviCvor->broj = broj;
    noviCvor->sledeci = *glava;
    *glava = noviCvor;
}

```

```

void prikazi(PCVOR glava) {
    int brojac = 1;
    while (glava) {
        printf("%d. %d\n", brojac++, glava->broj);
        glava = glava->sledeci;
    }
}

int postoji_u_listi(PCVOR glava, int broj) {
    while (glava) {
        if (glava->broj == broj) {
            return 1;
        }
        glava = glava->sledeci;
    }
    return 0;
}

void unija(PCVOR lista1, PCVOR lista2, PCVOR* lista3) {
    while (lista1) {
        if (postoji_u_listi(*lista3, lista1->broj) == 0) {
            dodaj(lista3, lista1->broj);
        }
        lista1 = lista1->sledeci;
    }
    while (lista2) {
        if (postoji_u_listi(*lista3, lista2->broj) == 0) {
            dodaj(lista3, lista2->broj);
        }
        lista2 = lista2->sledeci;
    }
}

void velika_u_mala(char rec[]) {
    for (int i = 0; rec[i] != '\0'; i++) {
        rec[i] = tolower(rec[i]);
    }
}

int postoji(char* naziv_datoteke, char* rec) {
    FILE* datoteka = fopen(naziv_datoteke, "r");
    if (datoteka == NULL) {
        printf("Greska prilikom otvaranja datoteka.\n");
        return;
    }

    char trenutna_reč[100];
    while (fscanf(datoteka, "%s", trenutna_reč) == 1) {
        velika_u_mala(trenutna_reč);
        if (strcmp(trenutna_reč, rec) == 0) {
            fclose(datoteka);
            return 1;
        }
    }

    fclose(datoteka);
    return 0;
}

```



```

//POMOCNE FUNKCIJE

void ispisMatrice(int mat[][20], int n, int m) {
    printf("Matrica: \n");
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            printf("%2d ", mat[i][j]);
        }
        printf("\n");
    }
    printf("\n");
}

void ispisNiza(int niz[], int n) {
    printf("Niz: \n");
    for (int i = 0; i < n; i++) {
        printf("%d ", niz[i]);
    }
    printf("\n\n");
}

int main() {

    //01. Nuliranje matrice
    int mat[5][20] = {
        {1, 2, 3, 4, 5},
        {6, 7, 8, 9, 10},
        {11, 0, 13, 14, 0},
        {16, 17, 18, 19, 20},
        {21, 22, 23, 24, 25}
    };
    nuliraj(mat, 5, 5);
    ispisMatrice(mat, 5, 5);

    //02. Izbacivanje duplikata iz matrice
    int niz[] = { 5, 8, 6, 4, 1, 2, 5, 8, 5, 4, 1 };
    int n = sizeof(niz) / sizeof(niz[0]);
    izbaci_duplikate(niz, &n);
    ispisNiza(niz, n);

    //03.00. Definisanje promenljivih
    PCVOR lista1 = NULL, lista2 = NULL, lista3 = NULL;
    int vrednostiLista1[] = { 1, 5, 2, 8 };
    int vrednostiLista2[] = { 3, 1, 6, 8, 9 };

    //03.01. Ucitavanje vrednosti u listu1
    int velicinaLista1 = sizeof(vrednostiLista1) /
sizeof(vrednostiLista1[0]);
    for (int i = 0; i < velicinaLista1; i++) {
        dodaj(&lista1, vrednostiLista1[i]);
    }

    //03.02. Ucitavanje vrednosti u listu2
    int velicinaLista2 = sizeof(vrednostiLista2) /
sizeof(vrednostiLista2[0]);
    for (int i = 0; i < velicinaLista2; i++) {
        dodaj(&lista2, vrednostiLista2[i]);
    }
}

```

```
//03.03. Racunanje unije i prikaz liste
unija(lista1, lista2, &lista3);
prikazi(lista3);

//04. Prebacivanje velikih u mala slova
char rec[] = "vEliKA";
velika_u_mala(rec);
printf("\nNakon konvertovanja: %s\n", rec);

//05. Provera sadrzaja datoteke
int rezultat = postoji("datoteka.txt", "lorem");
printf("\nPovratna vrednost funkcije: %d\n", rezultat);

return 0;
}
```