JMBAG

IME I PREZIME

Programiranje 2 – 1. pismeni ispit, 27. 6. 2025.

Upute: Na skraćenom ispitu je dozvoljeno koristiti samo pribor za pisanje i brisanje, te službeni podsjetnik. Kalkulatori, razne neslužbene tablice, papiri i sl., nisu dozvoljeni! **Mobitele isključite i spremite!** Sva rješenja napišite isključivo na papire sa zadacima, jer jedino njih predajete. Obavezno predajte sve papire sa zadacima, čak i ako neke zadatke niste rješavali. Ne zaboravite se **potpisati** na svim papirima! Skice smijete raditi i na drugim papirima koje će vam dati dežurni asistent.

Zadatak 1 (25 bodova) Neka je dano cjelobrojno dvodimenzionalno polje naziva `Visina`, koje se sastoji od m redaka i n stupaca te je ispunjeno različitim nenegativnim vrijednostima. Napišite funkciju koja najprije dinamički alocira novo dvodimenzionalno polje `Rezultat` istih dimenzija te postavlja njegove vrijednosti identično kao u polju `Visina` (kreira novi objekt koji je kopija staroga). Funkcija, uz originalno polje i dimenzije polja, prima i indekse $i \in \{0, \ldots, m-1\}$, $j \in \{0, \ldots, n-1\}$ te zatim mijenja vrijednosti u polju `Rezultat` tako da, počevši od pozicije `Rezultat[i][j]`, „poplavljuje“ sve susjedne ćelije (gore, dolje, lijevo, desno) koje imaju istu vrijednost kao početna ćelija. Vrijednosti svih „poplavljenih“ ćelija pritom se zamjenjuju s -1. Funkcija vraća pokazivač na dvodimenzionalno polje `Rezultat`. Napišite pomoćnu funkcija koja oslobađa memoriju zauzetu od funkcije `Flood`.

Primjer:

Početna pozicija: $(i, j) = (2, 1)$

Visina

$$\begin{bmatrix} 0 & 1 & 1 & 0 & 2 & 2 \\ 1 & 0 & 1 & 0 & 2 & 2 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 2 & 0 & 1 & 1 & 1 & 0 \\ 2 & 2 & 1 & 0 & 0 & 0 \end{bmatrix}$$

Rezultat

$$\begin{bmatrix} 0 & 1 & 1 & -1 & 2 & 2 \\ 1 & -1 & 1 & -1 & 2 & 2 \\ -1 & -1 & -1 & -1 & 1 & 1 \\ 2 & -1 & 1 & 1 & 1 & 0 \\ 2 & 2 & 1 & 0 & 0 & 0 \end{bmatrix}$$

```c
#include <stdlib.h>
int **alociraj_i_kopiraj(int **Visina, int m, int n) {
    int **Rezultat = malloc(m * sizeof(int *));
    for (int i = 0; i < m; i++) {
        Rezultat[i] = malloc(n * sizeof(int));
        for (int j = 0; j < n; j++) {
            Rezultat[i][j] = Visina[i][j];
        }
    }
    return Rezultat;
}
void poplava(int **matrica, int m, int n, int i, int j, int pocetna_vrijednost) {
    // Provjera granica i vrijednosti
    if (i < 0 || i >= m || j < 0 || j >= n)
        return;
    if (matrica[i][j] != pocetna_vrijednost)
        return;

    // Označi kao "poplavljeno"
    matrica[i][j] = -1;

    // Rekurzivno poplavi susjede (gore, dolje, lijevo, desno)
    poplava(matrica, m, n, i - 1, j, pocetna_vrijednost); // gore
    poplava(matrica, m, n, i + 1, j, pocetna_vrijednost); // dolje
    poplava(matrica, m, n, i, j - 1, pocetna_vrijednost); // lijevo
    poplava(matrica, m, n, i, j + 1, pocetna_vrijednost); // desno
}
int **Flood(int **Visina, int m, int n, int i, int j) {
    int **Rezultat = alociraj_i_kopiraj(Visina, m, n);
    int pocetna_vrijednost = Rezultat[i][j];
    poplava(Rezultat, m, n, i, j, pocetna_vrijednost);
    return Rezultat;
}
void oslobodi(int **matrica, int m) {
    for (int i = 0; i < m; i++)
        free(matrica[i]);
    free(matrica);
}
```

JMBAG

IME I PREZIME

Programiranje 2 – 1. pismeni ispit, 27. 6. 2025.

Zadatak 2 (25 bodova) Napišite implementaciju funkcije s prototipom

```
int involutorna(FILE* in)
```

koja kao argument prima pokazivač na tekstualnu datoteku otvorenu za čitanje u kojoj je zapisana kvadratna matrica sa cjelobrojnim koeficijentima. Matrica je zapisana redak po redak u linijama datoteke te su elementi retka odvojeni razmacima. Funkcija vraća 1 ako je matrica iz datoteke na koju pokazuje `in` involutorna, a inače vraća 0. Podsjetimo se, matrica je involutorna ako je sama sebi inverz.

Primjer: Za pokazivač `in` koji pokazuje na tekstualnu datoteku sa sadržajem

```
2 -1
3 -2
```

funkcija `involutorna` treba vratiti 1 jer vrijedi

$$\begin{pmatrix} 2 & -1 \\ 3 & -2 \end{pmatrix} \cdot \begin{pmatrix} 2 & -1 \\ 3 & -2 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

```c
#include <stdio.h>

int involutorna(FILE* in) {

    int n = 0;
    char c;

    // Brojanje razmaka u prvom retku za određivanje dimenzije
    while (fscanf(in, "%c", &c) == 1 && c != '\n')
        if (c == ' ') n++;
    n++; // broj elemenata je broj razmaka + 1

    int matrica[n][n];
    int rezultat[n][n];

    rewind(in);  // vrati se na početak datoteke

    // Učitaj matricu dimenzije n x n
    for (int i = 0; i < n; i++)
        for (int j = 0; j < n; j++)
            if (fscanf(in, "%d", &matrica[i][j]) != 1)
                return 0; // greška u čitanju

    // Pomnoži matricu samu sa sobom: rezultat = matrica * matrica
    for (int i = 0; i < n; i++)
        for (int j = 0; j < n; j++) {
            rezultat[i][j] = 0;
            for (int k = 0; k < n; k++)
                rezultat[i][j] += matrica[i][k] * matrica[k][j];
        }

    // Provjeri je li rezultat identična matrica (jedinice na dijagonali, nule drugdje)
    for (int i = 0; i < n; i++)
        for (int j = 0; j < n; j++)
            if ((i == j && rezultat[i][j] != 1) || (i != j && rezultat[i][j] != 0))
                return 0;

    return 1;
}
```

JMBAG

IME I PREZIME

Programiranje 2 – 1. pismeni ispit, 27. 6. 2025.

Zadatak 3 (5+10+10 bodova) Na jednom sveučilištu studenti imaju pravo na subvencioniranu prehranu. Podaci o subvenciji nalaze se na tzv. "iksici", koja sadrži identifikacijski broj studenta - JMBAG (string duljine 10) te iznos subvencije za troškove prehrane na koji student ima pravo (nenegativan realan broj). Prilikom korištenja iksice, 70% iznosa računa podmiruje se sredstvima s iksice (tada se sredstva na iksici umanjuju za taj iskorišteni iznos). Jelovnik u restoranu predstavljen je kao niz jela, pri čemu svako jelo ima svoje ime (string duljine najviše 30) te cijenu (pozitivan realan broj). Svaki restoran bilježi promet tijekom jednog dana u obliku niza računa. Račun sadrži broj odabranih jela (prirodan broj), popis odabranih jela (niz jela), vrijeme kupnje zapisano u formatu `hh:mm` (string duljine 5) te ukupan iznos računa (nenegativan realan broj).

(a) Definirajte tipove `iksica`, `jelo` i `racun`, tako da bude moguća deklaracija `iksica x`, `jelo j` i `racun r`.

(b) Napišite funkciju

```
racun *provuci_iksicu(iksica *x, char vrijeme[], jelo jelovnik[], int m, char* odabir[], int k)
```

koja prima pokazivač na iksicu, string duljine 5 u kojem je zapisano vrijeme u formatu `hh:mm`, jelovnik duljine m te niz stringova duljine k. Ti stringovi predstavljaju imena odabranih jela s jelovnika. Funkcija izrađuje novi račun i vraća pokazivač na njega te ispisuje: `Za platiti:x` gdje je `x` iznos koji **student treba platiti** (nakon uračunate subvencije). Funkcija također treba smanjiti iznos subvencije na iksici za 70% iznosa računa, kako je prethodno opisano. Ako student **nema dovoljno sredstava** na iksici, funkcija vraća `NULL` i ispisuje: `ODBIJENO`.

(c) Napišite funkciju `void izbirljivi(racun restoran[], int n)` koja prima niz računa restorana duljine n tijekom jednog dana i sortira ga uzlazno prema vremenu izdavanja. Složenost sortiranja mora biti $\mathcal{O}(n \log n)$, ako je složenost $\mathcal{O}(n^2)$ ili veća nosi podzadatak nosi 0 bodova.

Napomene:
Zabranjeno je korištenje dodatnih nizova. U (b) zadatku možete alocirati memoriju samo za potrebu izrade računa. Ne treba provjeravati uspješnost (re)alokacije. Uz standardnu biblioteku `stdio.h`, u ovome zadatku dozvoljeno je korištenje biblioteke `stdlib.h` i `string.h`.

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

typedef struct{
    char JMBAG[11];
    double subvencija;
} iksica;

typedef struct{
    char ime[31];
    double cijena;
} jelo;

typedef struct{
    int br_jela;
    jelo *popis;
    char vrijeme[6];
    double iznos;
} racun;

int comp(const void *a, const void *b){
    racun r1 = *((racun*)a);
    racun r2 = *((racun*)b);
    return strcmp(r1.vrijeme, r2.vrijeme);
}
```

```
racun *provuci_iksicu(iksica *x, char vrijeme[], jelo jelovnik[], int m, char* odabir[], int k){

    racun *novi = (racun*)malloc(sizeof(racun));
    novi->br_jela = k;
    strcpy(novi->vrijeme, vrijeme);

    // Ostaje dodati jela i iznos racuna.
    jelo *p = novi->popis;
    p = NULL;
    double iznos = 0.0;
    int i, j;
    for(i = 0; i < k; i++){
        p = realloc(p, (i+1)*sizeof(jelo));
        for(j = 0; j < m; j++){
            if(!strcmp(jelovnik[j].ime, odabir[i])){
                p[i] = jelovnik[j];
                iznos += jelovnik[j].cijena;
            }
        }
    }
    novi->popis = p;
    novi->iznos = iznos;

    // Ako nema dovoljno subvencije.
    if(0.7*iznos > x->subvencija){
        free(p);
        free(novi);
        printf("ODBIJENO\n");
        return NULL;
    }

    x->subvencija -= 0.7*iznos;
    printf("Za platiti: %.2f\n", 0.3*iznos);
    return novi;
}

void vrijeme_sort(racun restoran[], int n){
    qsort(restoran, n, sizeof(racun), comp);
    return;
}
```

JMBAG

IME I PREZIME

Programiranje 2 – 1. pismeni ispit, 27. 6. 2025.

Zadatak 4 (4+9+12 bodova) Sve pjesme koje pripadaju popisu za reproduciranje (playlisti) su spremljene u vezanoj listi. Za svaku pjesmu pamtimo naziv (string s najviše 20 znakova), izvođača (string s najviše 30 znakova), trajanje u sekundama (cijeli broj) i je li označena kao omiljena (cijeli broj koji ima vrijednost 1 ako je i 0 ako nije).

(a) Napišite deklaraciju odgovarajućeg tipa podataka za pjesmu na način da bude moguće definirati varijablu naredbom `pjesma p;`. Definirajte samo podatke koji su nužni za čuvanje takve liste u memoriji.

(b) Napišite funkciju `pjesma* izdvoji_omiljene(pjesma* playlist)` koja prima pokazivač na listu pjesama. Funkcija stvara novu vezanu listu koja sadrži sve pjesme iz liste `playlist` koje su označene kao omiljene i vraća pokazivač na početak novostvorene liste, a ako nema takvih pjesama onda vraća `NULL`. Za elemente nove liste je potrebno alocirati memoriju, te izvorna lista `playlist` ne smije biti promijenjena.

(c) Napišite funkciju `void premjesti_kratke(pjesma** playlist, pjesma** playlist_kratke, int max_sec)`. Funkcija prima dvostruki pokazivač na popis pjesama `playlist`, dvostruki pokazivač `playlist_kratke` preko kojeg će se vratiti lista kratkih pjesama, te `max_sec` koji definira maksimalno trajanje u sekundama za koje se pjesma smatra *kratkom* (kratka je ako traje `<= max_sec`). Lista na koju pokazuje `*playlist_kratke` inicijalno je prazna (`NULL`) i funkcija je popunjava.

Funkcija treba premjestiti sve kratke pjesme iz liste `*playlist` u listu na koju pokazuje `*playlist_kratke`. Premještanje treba izvršiti **bez alociranja dodatne memorije**, tj. preslagivanjem postojećih čvorova. Redoslijed pjesama u listi `*playlist_kratke` mora biti isti kao u izvornoj listi.

Napomena:

Dozvoljeno je korištenje pomoćnih funkcija. Nije dozvoljeno korištenje pomoćnih nizova. Možete pretpostaviti da su sve ulazne liste ispravno formirane. Jedine dozvoljene biblioteke u ovom zadatku su `stdlib.h` i `string.h`.

Primjer:

- `moja_playlista -> {"HeyJude", "TheBeatles", 431, 1} -> {"Stairway", "LedZeppelin", 482, 0} -> {"Imagine", "JohnLennon", 183, 1} -> {"Yesterday", "TheBeatles", 125, 0} -> {"LightMyFire", "TheDoors", 428, 1} -> NULL`

```
pjesma* omiljene = izdvoji_omiljene(moja_playlista);
```

- `omiljene->{"HeyJude", "TheBeatles", 431, 1} -> {"Imagine", "JohnLennon", 183, 1} -> {"LightMyFire", "TheDoors", 428, 1} -> NULL`

```
premjesti_kratke(&moja_playlista, &lista_kratkih, 200);
```

- `lista_kratkih->{"Imagine", "JohnLennon", 183, 1} -> {"Yesterday", "TheBeatles", 125, 0} -> NULL`
- `moja_playlista->{"HeyJude", "TheBeatles", 431, 1} -> {"Stairway", "LedZeppelin", 482, 0} -> {"LightMyFire", "TheDoors", 428, 1} -> NULL`

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

typedef struct _pjesma
{
    char naziv[21];
    char izvodac[31];
    int trajanje;
    int omiljena;

    struct _pjesma *next;
} pjesma;

pjesma *izdvoji_omiljene(pjesma *playlist)
{
    if (playlist == NULL)
        return NULL;

    pjesma *ret_first = NULL;
    pjesma *ret_last = NULL;
```

```
    for (pjesma *p = playlist; p != NULL; p = p->next) {
        if (p->omiljena) {
            pjesma *nova = (pjesma *) malloc(sizeof(pjesma));
            strcpy(nova->naziv, p->naziv);
            strcpy(nova->izvodac, p->izvodac);
            nova->trajanje = p->trajanje;
            nova->omiljena = p->omiljena;
            nova->next = NULL;

            if (ret_last == NULL) {
                // ret lista je trenutno prazna
                ret_first = ret_last = nova;
            } else {
                ret_last->next = nova;
                ret_last = nova;
            }
        }
    }

    return ret_first;
}

void premjesti_kratke(pjesma **playlist, pjesma **playlist_kratke, int max_sec)
{
    pjesma *kratke_first = NULL;
    pjesma *kratke_last = NULL;

    // Iteriramo po playlist
    pjesma *prev = NULL;
    pjesma *p = *playlist;
    while (p) {
        if (p->trajanje > max_sec) {
            // Nije kratka, ne moramo nista raditi s pjesmom p
            // nastavljamo s iteriranjem
            prev = p;
            p = prev->next;
            continue;
        }

        // Moramo premjestiti pjesmu p na kraj liste kratkih
        pjesma *tmp = p->next;
        p->next = NULL;
        if (kratke_first == NULL) {
            kratke_first = kratke_last = p;
        } else {
            kratke_last->next = p;
            kratke_last = p;
        }

        p = tmp;
        if (prev == NULL) {
            // Prvi element se promijenio
            *playlist = p;
        } else {
            prev->next = p;
        }
    }

    *playlist_kratke = kratke_first;
}
```