# Algoritmi i strukture podataka
## 1. međuispit
### 4.4.2006.

Napomene za sve zadatke:

- Nije dopušteno korištenje goto naredbe, te statičkih i globalnih varijabli.
- U svakom zadatku u kojem se koristi struktura, strukturu treba i definirati

**1.** *(4 boda)* Jedan zapis datoteke organizirane po principu raspršenog adresiranja sadrži **matični broj studenta** (int), **ime i prezime** (50+1 znak), **godinu na kojoj trenutno studira** (int) te **trenutni prosjek ocjena** (float). Prazni se zapis prepoznaje po matičnom broju jednakom 0. Veličina bloka na disku je 2048. Očekuje se najviše 120000 zapisa, a tablica je dimenzionirana za 25% veći kapacitet. Kod upisa se primjenjuje metoda cikličkog preljeva. Ključ zapisa je matični broj, a transformacija ključa u adresu se obavlja zadanom funkcijom

```
int adresa(int matbr);
```

Napisati funkciju koja će pronaći i vratiti zapis o studentu sa zadanim matičnim brojem. Funkcija preko imena treba vratiti 1 ako se zapis nalazi u „svom“ pretincu, 2 ako je zapisan kao preljev, a 0 ako zapis ne postoji u datoteci. Funkcija treba imati prototip:

```
int fun(FILE *f, int matbr, zapis *student);
```

```
#define N 120000
#define BLOK 2048
#define C BLOK/sizeof(zapis)
#define M (int)(N*1.25/C)

typedef struct z {
    int matbr;
    char ipr[50+1];
    int godina;
    float prosjek;
} zapis;

int fun(FILE *f, int matbr, zapis *student) {
    zapis pretinac[C];
    int i, j, poc;

    i = poc = adresa(matbr);
    do {
        fseek(f, i*BLOK, SEEK_SET);
        fread(pretinac, sizeof(zapis), C, f);
        for (j=0; j<C; j++) {
            if (pretinac[j].matbr == 0) return 0; // ako nema brisanja
            if (pretinac[j].matbr == matbr) {
                *student = pretinac[j];
                if (i == poc) return 1;          // u svom pretincu
                else return 2;                   // preljev
            }
        }
        i = (i+1)%M;
    }while (i!=poc);

    return 0;    // zapis ne postoji
}
```

**OKRENI**

**2.** *(4 boda)* Funkcija koja u sortiranom polju cijelih brojeva (**int**) traži zadani element i ima složenost $O(\log_2 n)$ glasi:

```
int BinTraz (tip a[], int x, int n) {
  int donji, srednji, gornji;
  donji = 0; gornji = n - 1;
  while (donji <= gornji) {
    srednji = (donji + gornji) / 2; // "prepolovi" (pod)polje
    if (a [srednji] < x)
      donji = srednji + 1;          // trazeni u desnom dijelu
    else if (a [srednji] > x)
      gornji = srednji - 1;         // trazeni u lijevom dijelu
    else
      return srednji;               // nadjen
  }
  return -1;                        // nije nadjen
}
```

Napisati novu REKURZIVNU funkciju `BinTraz2`, koja će imati isti prototip, istu složenost i raditi isto što i gornja funkcija.

```
int BinTraz2 (int a[], int x, int n) {
    int srednji;
    int pom;

    if (n == 0) return -1; // zapis ne postoji
    srednji = (n-1)/2;
    if (a[srednji] > x) return BinTraz2(a, x, srednji);
    else if (a[srednji] < x) {
        pom = BinTraz2(a+srednji+1, x, n-srednji-1);
        if (pom == -1) return -1;
        else return srednji + 1 + pom;
    }
    else return srednji;
}
```

**3.** *(4 boda)* Napisati funkciju koja će napraviti i vratiti duplikat zadanog polja realnih brojeva. Memoriju za duplikat potrebno je zauzeti unutar funkcije. Funkcija mora imati prototip:

```
float *duplikat(float *polje, int n);
```

Odrediti apriornu složenost funkcije za najgori, najbolji i prosječni slučaj u ovisnosti o broju elemenata polja.

```
float *duplikat(float *polje, int n) {
    float *polje2;
    int i;

    polje2 = (float *)malloc(n*sizeof(float));
    if (polje2) {
        for (i=0; i<n; i++) polje2[i] = polje[i];
    }

    return polje2;
}
```

Sve složenosti su O(n)

**OKRENI**

**4.** *(3 boda)* Dana su dva odsječka programa. Svaki od njih traži zadani zapis na glavnoj dijagonali kvadratnog 2D polja:

**I.**

```
nasao = 0;
for (i=0; i<n; i++) {
   if (polje[i][i] == zadani) {
      nasao = 1;
      break;
   }
}
```

**II.**

```
nasao = 0;
for (i=0; i<n; i++) {
   for (j=0; j<n; j++) {
      if (i==j && polje[i][j]==zadani)
         nasao = 1;
   }
}
```

Odrediti apriornu složenost oba odsječka za najbolji, najgori i prosječni slučaj u ovisnosti o broju redaka 2D polja (**n**). Detaljno objasniti svoje odgovore.

**I**
Najbolji: O(1)
Najgori: O(n)
Prosječni: O(n)

**II**
Najbolji: $O(n^2)$
Najgori: $O(n^2)$
Prosječni: $O(n^2)$

**OKRENI**