JMBAG **IME I PREZIME**

Algoritmi i strukture podataka –dekanski ispitni rok

17.rujna 2014.

*Nije dopušteno korištenje globalnih i statičkih varijabli te naredbe **goto**. Neefikasna rješenja mogu donijeti manje bodova. Nerekurzivne funkcije se ne priznaju kao rješenja u zadacima u kojima se traži rekurzivna funkcija i obratno.*
*Ispit nosi maksimalno 70 bodova, a prag za prolaz pismenog ispita je **35** bodova. 3. zadatak rješavate na ovom obrascu dok ostale zadatke rješavate na svojim listovima papira. Ovaj obrazac morate predati.*

Zadatak 1. (20 bodova)

Svaki zapis datoteke organizirane po načelu raspršenog adresiranja sadrži podatke o jednom proizvodu i definiran je strukturom:

```
typedef struct{
   int sifra;
   char naziv[50+1];
   double cijena;
} zapis;
```

Šifra nula (0) označava prazan zapis. Domena šifri je određena tipom podataka, a cijene su garantirano manje od 10^8. Veličina bloka na disku je definirana simboličkom konstantom BLOK. Očekuje se najviše 2500000 zapisa, a kapacitet tablice je 20% veći. Prilikom upisa primjenjuje se metoda cikličkog preljeva. Ključ zapisa je *sifra*, a pretvorba ključa u adresu se obavlja već pripremljenom funkcijom prototipa:

```
int adresa(int sifra);
```

Napišite funkciju `ispis` koja će ispisati šifre, nazive i cijene onih proizvoda koji su završili u preljevu i čija je cijena veća od prosječne cijene svih valjanih zapisa u tom pretincu. Format ispisa je unaprijed zadan primjerom (prvi red označava znamenku desetica, a drugi znamenku jedinica rednog broja kolone znakovnog sučelja):

```
         1         2         3         4         5         6         7
1234567890123456789012345678901234567890123456789012345678901234567890123
     10000 Crvena kemijska za ispravljanje ispita                  99.90
```

Prototip funkcije zadan je s:

```
void ispis(FILE *fi);
```

Zadatak 2. (10 bodova)

Napisati rekurzivnu funkciju koja će zrcalno obrnuti binarno stablo, tako da korijen stabla ostane korijen, a lijevo i desno podstablo zamijene mjesta u svakoj razini. Neka je čvor stabla zadan tipom `cvor`:

```
typedef struct st_cvor {
       KORISNICKI_TIP vrijednost;
       struct st_cvor *l, *d;
} cvor;
```

Zadatak 3. (12 bodova)

a) Koja je apriorna složenost zbrajanja dvaju n-znamenkastih brojeva na papiru?

b) Poredajte uzlazno sljedeće složenosti (nakon što su poredani, među izrazima mora biti znak **>**,**<** ili **=**):
$O(2^n)$, $O(n^{1.5})$, $O(n)$, $O(n \log_2 n)$, $O(n \ln n)$, $O(\sqrt{n})$, $O(n!)$, $O(n^n)$, $O(n^{1000})$, $O(3^n)$

c) Koja je apriorna složenost posljednjih triju linija sljedećeg programskog odsječka:

```
double n;
/* ... odsječak u kojem se n postavlja
na neku vrijednost ...*/
while (n>1) {
       n*=0.999;
}
```

Zadatak 4. (18 bodova)

Binarno stablo sadrži pozitivne cijele brojeve, a čvor je definiran strukturom:

```
typedef struct st_cvor {
       int vrijednost;
       struct st_cvor *lijevo, *desno;
} cvor;
```

Treba napisati rekurzivnu funkciju koja će vratiti koliko unutarnjih čvorova u stablu ima veći prosjek elemenata u lijevom podstablu nego u desnom podstablu i koliko unutarnjih čvorova u stablu ima veći prosjek elemenata u desnom podstablu nego u lijevom.

Zadatak 5. (10 bodova)

U polje cijelih brojeva pohranjen je sljedeći niz brojeva:

16, 20, 6, 22, 5, 8, 17, 21, 23, 3, 2, 19.

a) Ilustrirajte (napišite sadržaj polja nakon svake promjene) stvaranje gomile s relacijom ***veći od*** (max heap) od zadanog polja brojeva algoritmom čija je složenost za najgori slučaj *O(n).*
b) Počevši od gomile kreirane u podzadatku a), prikažite sortiranje zadanog niza *heapsortom*, prikazujući svaki korak sortiranja (napišite sadržaj polja nakon svake promjene).

Rješenja:

Zadatak 1. (20)

```
#define N 250000
#define C (BLOK/sizeof (zapis))
#define M ((int)(N*1.2/C))

void ispis(FILE *fi) {
  zapis pretinac[C];
  int i, j;
  int br_zapisa = 0;
  double suma = 0;
  double prosjecna_cijena = 0;

  for (i = 0; i < M; i++) {
    fseek (fi, i*BLOK, SEEK_SET);
    fread (pretinac, sizeof (pretinac), 1, fi);
    suma = 0; br_zapisa = 0;
    for (j = 0; j < C; j++) {
      if (pretinac[j].sifra) { //preskačemo prazne zapise
        br_zapisa++;
        suma += pretinac[j].cijena;
      }
    }
    prosjecna_cijena = suma / br_zapisa;
    for (j = 0; j < C; j++) {
      /*preskacemo obrisane zapise, zapise koji nisu u preljevu i zapise cijene manje od prosjeka*/
      if(pretinac[j].sifra != 0 && pretinac[j].cijena > suma && adresa(pretinac[j].sifra) != i) {
        printf( "%10d %-50s %11.2f\n", pretinac[j].sifra, pretinac[j].naziv, pretinac[j].cijena);
      }
    }
  }
}
/* primjetiti: bolje je prvo raditi usporedbu cijene i sume (poznato i kratko vrijeme izvodjenja)
nego  pozivati funkciju koja moze i ne mora biti banalne izvedbe i brzog vremena izvodjenja */
```

Zadatak 2. (10)

```
void okreni( cvor *korijen ) {
       cvor *temp;
       if( korijen ) {
              okreni( korijen -> lijevo );
              okreni( korijen -> desno );
              temp = korijen -> lijevo;
              korijen -> lijevo = korijen -> desno;
              korijen -> desno = temp;
       }
}
```

Zadatak 3. (12)

a) O(n)

b) $O(\sqrt{n}) < O(n) < O(n \log_2 n) = O(n \ln n) < O(n^{1..5}) < O(n^{1000}) < O(2^n) < O(3^n) < O(n!) < O(n^n)$

c) O(log n)

Zadatak 4. (18)

```
void zaProsjek (cvor *korijen, int *broj, int *zbroj) {
  if (korijen != NULL) {
        zaProsjek(korijen->lijevo,broj,zbroj);
        zaProsjek(korijen->desno,broj,zbroj);
     (*broj)++;
        (*zbroj)+=korijen->vrijednost;
  } else {
    return;
  }
}

void prebrojiStanje(cvor *korijen, int *sL, int *sD){
    int zbroj, broj;
    float prosjekL, prosjekD;

    if (korijen==NULL) return;
    // pogledaj rekurzivno lijevo i desno podstablo
    prebrojiStanje(korijen->lijevo,sL,sD);
    prebrojiStanje(korijen->desno,sL,sD);
    // sada odredi je li veći prosjek lijevo ili desno
    broj=0; zbroj=0;
    zaProsjek(korijen->lijevo, &broj, & zbroj);
    if (broj!=0){
        prosjekL=((double) zbroj)/broj;
    }
    else prosjekL=0; // signal da nema elemenata u podstablu

    broj=0; zbroj=0;
    zaProsjek(korijen->desno, &broj, & zbroj);
    if (broj!=0){
           prosjekD=((double) zbroj)/broj;
    }
    else prosjekD=0;

    if (prosjekL > prosjekD) {
        (*sL)++;
    }
    else if(prosjekD > prosjekL){
        (*sD)++;
    }
}
```

Zadatak 5. (10)

16	20	6	22	5	**8**	17	21	23	3	2	**19**
16	20	6	**22**	5	19	17	21	**23**	3	2	8
16	20	**6**	23	5	**19**	17	21	22	3	2	8
16	20	19	23	5	**6**	17	21	22	3	2	**8**
16	**20**	19	**23**	5	8	17	21	22	3	2	6
16	23	19	**20**	5	8	17	21	**22**	3	2	6
16	**23**	19	22	5	8	17	21	20	3	2	6
23	**16**	19	**22**	5	8	17	21	20	3	2	6
23	22	19	**16**	5	8	17	**21**	20	3	2	6
23	22	19	21	5	8	17	16	20	3	2	6

23	22	19	21	5	8	17	16	20	3	2	6
6	22	19	21	5	8	17	16	20	3	2	**23**
22	6	19	21	5	8	17	16	20	3	2	**23**
22	21	19	6	5	8	17	16	20	3	2	**23**
22	21	19	20	5	8	17	16	6	3	2	**23**
2	21	19	20	5	8	17	16	6	3	**22**	**23**
21	2	19	20	5	8	17	16	6	3	**22**	**23**
21	20	19	2	5	8	17	16	6	3	**22**	**23**
21	20	19	16	5	8	17	2	6	3	**22**	**23**
3	20	19	16	5	8	17	2	6	**21**	**22**	**23**
20	3	19	16	5	8	17	2	6	**21**	**22**	**23**
20	16	19	3	5	8	17	2	6	**21**	**22**	**23**
20	16	19	6	5	8	17	2	3	**21**	**22**	**23**
3	16	19	6	5	8	17	2	**20**	**21**	**22**	**23**
19	16	3	6	5	8	17	2	**20**	**21**	**22**	**23**
19	16	17	6	5	8	3	2	**20**	**21**	**22**	**23**
2	16	17	6	5	8	3	**19**	**20**	**21**	**22**	**23**
17	16	2	6	5	8	3	**19**	**20**	**21**	**22**	**23**
17	16	8	6	5	2	3	**19**	**20**	**21**	**22**	**23**
3	16	8	6	5	2	**17**	**19**	**20**	**21**	**22**	**23**
16	3	8	6	5	2	**17**	**19**	**20**	**21**	**22**	**23**
16	6	8	3	5	2	**17**	**19**	**20**	**21**	**22**	**23**
8	6	2	3	5	**16**	**17**	**19**	**20**	**21**	**22**	**23**
5	6	2	3	**8**	**16**	**17**	**19**	**20**	**21**	**22**	**23**
6	5	2	3	**8**	**16**	**17**	**19**	**20**	**21**	**22**	**23**
3	5	2	**6**	**8**	**16**	**17**	**19**	**20**	**21**	**22**	**23**
5	3	2	**6**	**8**	**16**	**17**	**19**	**20**	**21**	**22**	**23**
2	3	**5**	**6**	**8**	**16**	**17**	**19**	**20**	**21**	**22**	**23**
3	2	**5**	**6**	**8**	**16**	**17**	**19**	**20**	**21**	**22**	**23**
2	**3**	**5**	**6**	**8**	**16**	**17**	**19**	**20**	**21**	**22**	**23**