JMBAG **IME I PREZIME**

Algoritmi i strukture podataka –2. ispitni rok

2.rujna 2014.

*Nije dopušteno korištenje globalnih i statičkih varijabli te naredbe **goto**. Neefikasna rješenja mogu donjeti 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 se rješava na ovom obrascu, a ostali zadaci na vlastitim listovima papira. Ovaj obrazac se mora predati.*

Zadatak 1. (20 bodova)

Zadana je datoteka s podatcima o prodajnim artiklima organizirana po načelu raspšenog adresiranja. Svaki zapis se sastoji od šifre artikla (`long`), naziva artikla (do 50 znakova), kategorije (`short`), raspoložive količine (`int`), cijene (`float`) i težine (`float`). Ključ zapisa je šifra, a šifra 0 znači prazan zapis. Pretvorbu ključa u adresu obavlja već pripremljena funkcija prototipa:

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

Očekuje se do 1.000.000 zapisa, s time da je kapacitet datoteke 10% veći. Veličina bloka na disku je 4096 B. Za preljeve je rezervirano preljevno područje veličine četvrtine primarnog područja. Preljevno područje nalazi se na početku datoteke.

Potrebno je napisati funkciju prototipa

```
stavka *analiza( FILE *fi );
```

koja će napraviti analizu podataka iz datoteke i vratiti tražene informacije. Rezultat analize je polje struktura tipa `stavka`, pri čemu je `stavka` zadana odsječkom:

```
typedef struct st_stavka {
       short kategorija;
       float uk_vrijednost;
       float uk_tezina;
} stavka;
```

Funkcija vraća pokazivač na dinamički alocirano polje struktura s onoliko elemenata koliko je kategorija proizvoda u datoteci. Svaki element polja sadrži zbirne podatke za sve proizvode pojedine kategorije. Kategorije u datoteci ne moraju biti uzastopni brojevi, a polje ne mora biti sortirano po kategorijama.

Napomena: polje treba dinamički generirati prema broju kategorija (nije unaprijed poznato koliko je kategorija proizvoda u datoteci).

Zadatak 2. (14 bodova)

Jednostruko povezana lista sastoji se od atoma definiranih odsječkom:

```
typedef struct st_element {
       int vrijednost;
       struct st_element *sljed;
} element;
```

Napisati funkciju za dodavanje novog podatka (atoma) u listu tako da se očuva sortiranost liste. Funkcija prima pokazivač na pokazivač na korijen liste i podatak tipa `int`, a vraća 0 ili 1 ovisno o uspješnosti umetanja u listu.

Zadatak 3. (8 bodova)

Odredite gornju granicu (veliki O) složenosti i asimptotsku složenost zadanih odsječaka uz pretpostavku konstantne asimptotske složenosti funkcije f(x) i da je n >> 1. Obrazložite odgovor.

a)
```
a=0;
for(i=0;i<n;i++)
        for(j=n;j>i;j--)
                a+=f(j);
```

O:	asimptotska:

b)
```
a=0;
for(i=n; i>1; i=(int)sqrt(i))
        for(j=1;j<i;j*=2)
                a+=f(j);
```

O:	asimptotska:

Zadatak 4. (18 bodova)

Čvor binarnog stabla za pretraživanje je struktura:

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

Napisati rekurzivnu funkciju `nti` koja pronalazi n-ti po veličini član stabla. Prototip funkcije je:

```
int nti( int n, cvor *cv, int *rez )
```

pri čemu je `n` redni broj po veličini elementa kojeg tražimo (1 za najveći, 2 za drugi najveći itd.), a `rez` je lokacija na koju treba pohraniti rezultat. Pretpostaviti da je broj `n` uvijek manji od broja čvorova stabla.

Zadatak 5. (10 bodova)

U polje cijelih brojeva pohranjen je sljedeći niz brojeva:

5, 9, 2, 1, 8, 6, 7, 3, 4

a) **(5 bodova)** Ilustrirati uzlazno sortiranje algoritmom Bubblesort.
b) **(5 bodova)** Ilustrirati uzlazno sortiranje algoritmom Quicksort. Stožer za quicksort odabrati metodom aproksimacije medijana temeljem prvog, središnjeg i zadnjeg člana. Polja s manje ili točnocutoff = 3 elementa sortirati Insertion sortom.

Rješenja:

Zadatak 1. (20)

```
#define BLOK 4096
/* kapacitet pretinca: */
#define C ( BLOK / sizeof( artikl ) )
/* broj pretinaca u primarnom podrucju: */
#define M ( ( int ) ( 1.1 * N / C ) )
/* broj pretinaca u overflow podrucju: */
#define O ( ( int ) ( 0.25 * M ) )

typedef struct st_artikl {
       long sifra;
       char naziv[ 50 + 1 ];
       short kategorija;
       int kolicina;
       float cijena; float tezina;
} artikl;

stavka *analiza( FILE *fi ) {
  int i, j, k, b_stavki = 0, found = 0;
  stavka *stat = NULL, * temp;
  artikl pretinac[ C ];
  fseek( fi, 0L, SEEK_SET );
  for( i = 0; i < M + O; i ++ ) {
    fread( pretinac, sizeof( pretinac ), 1, fi );
    for( j = 0; j < C; j ++ ) {
      if( pretinac[j].sifra == 0 )
        continue;
      for( k = 0; k < b_stavki; k++ ) {
         if( stat[ k ].kategorija == pretinac[j].kategorija ) {
          found = 1;
          break;
        }
      }
      if( !found ) {
        stat = ( stavka * ) realloc( stat , ( b_stavki + 1 ) * sizeof( stavka ) );
        stat[ b_stavki ].kategorija = pretinac[ j ].kategorija;
        stat[ b_stavki ].vrijednost = pretinac[ j ].kolicina * pretinac[ j ].cijena;
        stat[ b_stavki ].uk_tezina = pretinac[ j ].kolicina * pretinac[ j ].tezina;
        b_stavki ++;
      }
      else {
        stat[ k ].vrijednost += pretinac[  j ].kolicina * pretinac[ j ].cijena;
        stat[ k ].uk_tezina += pretinac[ j ].kolicina * pretinac[ j ].tezina;
      }
    }
  }
  return stat;
}
```

Zadatak 2. (14)

```
int dodaj_element( element **korijen, int vrijednost ) {
       element *novi = NULL;
       if( ! ( novi = ( element * ) malloc( sizeof( element ) ) ) ) {
              return 0;
       }
       novi -> vrijednost = vrijednost;
       novi -> sljed = NULL;
       while( * korijen && ( ( * korijen ) -> vrijednost < vrijednost ) ) {
              korijen = & ( * korijen ) -> sljed;
       }
       novi -> sljed = * korijen;
       * korijen = novi;
       return 1;
}
```

Zadatak 3. (8)

a) broj iteracija je n + (n – 1) + (n – 2) + ... + 1, dakle:
$O(n^2)$;
$\Theta((n+1)*n/2)$

b) unutarnja petlja je logaritamske složenosti prema varijabli i, a i poprima po vrijednosti n, $\sqrt{n}$, $\sqrt{\sqrt{n}}$,... dakle:
$\Theta(\log_2 n^{1/2} + \log_2 n^{1/4} + \log_2 n^{1/8} + ...) = \Theta((1 + ½ + ¼ + \dots) * \log_2 n)) \sim= \Theta(2 \log_2 n)$
tj. $O(\log_2 n)$;

Zadatak 4. (18)

```
int nti( int n, cvor *cv, int *rez ) {
	int desno, lijevo;
	if( cv == NULL )
		return 0;
	desno = nti( n, cv->d, rez );

	if( desno == n - 1 ) {
		*rez = cv -> vrijednost;
	}
	lijevo = nti( n - desno – 1
		, cv -> l, rez );
	return desno + lijevo + 1;
}
```

Zadatak 5. (10)

a)

5 **9** **2** 1 8 6 7 3 4
5 2 **9** **1** 8 6 7 3 4
5 2 1 **9** **8** 6 7 3 4
5 2 1 8 **9** **6** 7 3 4
5 2 1 8 6 **9** **7** 3 4
5 2 1 8 6 7 **9** **3** 4
5 2 1 8 6 7 3 **9** **4**
5 **2** 1 8 6 7 3 4 9
2 **5** **1** 8 6 7 3 4 9
2 1 5 **8** **6** 7 3 4 9
2 1 5 6 **8** **7** 3 4 9
2 1 5 6 7 **8** **3** 4 9
2 1 5 6 7 3 **8** **4** 9
2 **1** 5 6 7 3 4 8 9
1 2 5 6 **7** **3** 4 8 9
1 2 5 6 3 **7** **4** 8 9
1 2 5 **6** **3** 4 7 8 9
1 2 5 3 **6** **4** 7 8 9
1 2 **5** **3** 4 6 7 8 9
1 2 3 **5** **4** 6 7 8 9
1 2 3 4 5 6 7 8 9

b)

5 9 2 1 **8** 6 7 3 **4**
4 9 2 1 **5** 6 7 **3** 8
4 **9** 2 1 **3** 6 7 5 8
4 3 2 1 **9** 6 7 **5** 8
4 **3** 2 **1** 5 6 7 9 8
1 **3** **2** 4 5 6 7 9 8
1 2 3 4 5 **6** **7** 9 **8**
1 2 3 4 5 6 **7** **9** 8
1 2 3 4 5 6 **9** **7** 8
1 2 3 4 5 6 7 **9** **8**
1 2 3 4 5 6 7 8 9