Algoritmi i strukture podataka –1. ispitni rok

16. rujna 2015.

*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.*
*Ispit nosi maksimalno 70 bodova, a prag za prolaz pismenog ispita je **35** bodova. Zadatak 4 rješava se na ispitnom primjerku, ostali zadaci na posebnim papirima.*

Zadatak 1. (22 boda)

Jedan zapis datoteke `studenti_jmbag.dat` organizirane po načelu raspršenog adresiranja definiran je strukturom

```
typedef struct stud {
      long jmbag;
      long oib;
      char ime_prezime[ 80 + 1 ];
} student;
```

Ključ zapisa u datoteci `studenti_jmbag.dat` je JMBAG (članska varijabla `jmbag`), a prazni zapisi označeni su nulom na mjestu šifre. Parametri za raspršeno adresiranje nalaze se u datoteci `parametri.h` i oni su:

- `BLOK .......... veličina bloka na disku`
- `C    .......... broj zapisa u jednom pretincu datoteke studenti_jmbag.dat`
- `M    .......... broj pretinaca u datoteci studenti_jmbag.dat`

Preljevi su realizirani ciklički upisom u prvi sljedeći slobodni pretinac. Transformacija ključa u adresu obavlja se zadanom funkcijom

```
int adresa_jmbag(long jmbag);
```

Napišite funkciju koja će sve neprazne zapise iz datoteke `studenti_jmbag.dat` prepisati u novu datoteku `studenti_oib.dat` koja je također organizirana po načelu raspršenog adresiranja, ali u kojoj je ključ zapisa OIB studenta (članska varijabla `oib`). U ovoj datoteci su preljevi također realizirani ciklički upisom u prvi sljedeći slobodni pretinac, a transformacija ključa u adresu obavlja se zadanom funkcijom

```
int adresa_oib(long oib);
```

Upis jednog zapisa u datoteku obavlja zadana funkcija

```
int upis(student *s, char *ime_datoteke);
```

koja prima adresu zapisa kojeg je potrebno upisati i ime datoteke u koju je potrebno upisati zapis, a kao povratnu vrijednost vraća broj pretinca u koji je zapis upisan.

Prototip tražene funkcije treba biti:

```
float prepisi();
```

Funkcija kao povratnu vrijednost treba vratiti omjer ukupnog broja preljeva u datoteci `studenti_jmbag.dat` i ukupnog broja zapisa. Pretpostaviti da je na početku izvođenja programa datoteka `studenti_oib.dat` prazna.

Zadatak 2. (18 bodova)

Napravite rekurzivnu funkciju `samoNeparni` čiji je prototip

```
long samoNeparni(long n);
```

Funkcija za zadani pozitivni cijeli broj `n > 0` vraća broj `k` koji se dobije tako da se iz broja `n` izostave parne znamenke

Npr: **`samoNeparni(2062) = 0, samoNeparni (21000) = 1, samoNeparni (123456) = 135.`**

Zadatak 3. (14 bodova)

Napisati funkciju koja će jednostruko povezanu linearnu listu proširiti novim elementima na način da između svaka dva čvora liste doda novi čvor čija će vrijednost biti jednaka sumi vrijednosti čvorova između kojih se dodaje. U obzir uzimati samo izvorne čvorove liste.

Prototip funkcije je

```
void prosiri (atom *glava);
```

Atom liste zadan je strukturom:

```
typedef struct at {
      int elem;
      struct at *sljed;
} atom;
```

Primjer: Peteročlana lista u nastavku (glava liste je prvi element slijeva) proširena je s dodatnim elementima koji su označeni masno (**bold**):

Originalna lista: 2 8 12 9 7

Proširena lista: 2 **10** 8 **20** 12 **21** 9 **16** 7

Zadatak 4. (6 bodova)

Koja je složenost sljedećeg odsječka u O i Ω notaciji u ovisnosti o argumentu n?

```
void funct(int n) {
  int i,j;
  for (i=n;i>0;--i) {
    for (j=1;j<i;j*=3) {
      printf("%d\n",j%2);
      j/=2;
    }
  }
}
```

O notacija (2 boda):

Ω notacija (1 bod):

```
void funct(int n) {
  int i,j;
  if (n<20)
    fact(n);
  else {
    if (n%2) {
      for (i=0;i<n;++i) {
        for (j=0;j<i;++j) {
          printf("%d\n",j%2);
        }
      }
    }
    else {
      for (i=0;i<n;++i) {
        printf("%d\n",i%2);
      }
    }
  }
}
```

O notacija (2 boda):

Ω notacija (1 bod):

(napomena: funkcija fact ima O(n!) i Ω(n!), a modulo operator u argumentu printf-a osigurava jednako vrijeme izvođenja printf-a bez obzira na veličinu argumenta)

Zadatak 5. (10 bodova)

U polje cijelih brojeva pohranjen je sljedeći niz brojeva:

8, 14, 6, 16, 13, 12, 15, 9

a) **(5 bodova)** Ilustrirajte (napišite sadržaj polja nakon svake promjene) stvaranja gomile s relacijom veći od (max heap) od zadanog polja brojeva algoritmom čija je složenost za najgori slučaj O(n).

b) **(5 bodova)** Počevši od gomile kreirane u podzadatku a), prikažite sortiranje zadanog niza heapsortom, prikazujući svaki korak sortiranja (napišite sadržaj polja nakon svake promjeren).

Rješenja:

Zadatak 1. (22 boda)

```
#include<stdlib.h>
#include<stdio.h>
#include "parametri.h"

float prepisi() {
        char ime_dat_jmbag[] = "studenti_jmbag.dat", ime_dat_oib[] = "studenti_oib.dat";
        FILE *fjmbag, *foib;
        student pretinac[ C ];
        int i, j, br_zapisa = 0, br_preljeva = 0, ocekivani_pretinac, upisani_pretinac;
        if( ( fjmbag = fopen( ime_dat_jmbag, "rb" ) ) == NULL ) {
                fprintf( stderr, "Ne mogu otvoriti datoteku %s za citanje\n", ime_dat_jmbag );
                exit( EXIT_FAILURE );
        }
        for( i = 0; i < M; i++ ) {
                fseek( fjmbag, i * BLOK, SEEK_SET );
                fread( pretinac, sizeof( pretinac ), 1, fjmbag );
                for( j = 0; j < C; j++ ) {
                        if( pretinac[ j ].jmbag ) {
                                br_zapisa ++;
                                ocekivani_pretinac = adresa_jmbag( pretinac[ j ].jmbag );
                                br_preljeva += ocekivani_pretinac != i;
                                upis( &pretinac[ j ], ime_dat_oib);
                        }
                }
        }
        fclose(fjmbag);
        return (float)br_preljeva/br_zapisa;
}
```

Zadatak 2. (18 bodova)

```
long samoNeparni(long n){
       long tmp;
       int zn;
       if (n<10){
              tmp=0;
              if (n % 2==1) tmp=n;
              return tmp;
       }
       zn=n % 10;
       tmp=samoNeparni(n/10);
       if(zn % 2==1) tmp=tmp*10+zn;
       return tmp;
}
```

Zadatak 3. (14 bodova)

```
void prosiri(atom *glava){
        atom *novi,*pom;
        while (glava){
                pom = glava->sljed;
                if (pom){
                        novi = (atom*)malloc(sizeof(atom));
                        novi->elem = glava->elem + pom->elem;
                        novi->sljed = pom;
                        glava->sljed = novi;
                }
                glava = pom;
        }
}
```

Zadatak 4. (6 bodova)

Unutarnja petlja algoritma je beskonačna te algoritam ne završava već je apriori složenost (i O i Ω) beskonačna!	$O(n^2)$, $\Omega(n)$ Funkcija fact ima faktorijelnu složenost, ali to je nevažno jer se funkcija poziva samo za male n. Za velike parne brojeve složenost određuju ugnježđene for petlje, a za neparne jednostavna for petlja. Najgori slučaj za velike brojeve je ako je broj je paran pa algoritam ima kvadratnu složenost, a najbolji slučaj je ako je broj neparan pa algoritam ima linearnu složenost.

Zadatak 5. (10 bodova)

```
 8, 14,  6, 16, 13, 12, 15,  9
 8, 14, 15, 16, 13, 12,  6,  9
 8, 16, 15, 14, 13, 12,  6,  9
16,  8, 15, 14, 13, 12,  6,  9
16, 14, 15,  8, 13, 12,  6,  9
16, 14, 15,  9, 13, 12,  6,  8

 8, 14, 15,  9, 13, 12,  6, 16
15, 14,  8,  9, 13, 12,  6, 16
15, 14, 12,  9, 13,  8,  6, 16
 6, 14, 12,  9, 13,  8, 15, 16
14,  6, 12,  9, 13,  8, 15, 16
14, 13, 12,  9,  6,  8, 15, 16
 8, 13, 12,  9,  6, 14, 15, 16
13,  8, 12,  9,  6, 14, 15, 16
13,  9, 12,  8,  6, 14, 15, 16
 6,  9, 12,  8, 13, 14, 15, 16
12,  9,  6,  8, 13, 14, 15, 16
 8,  9,  6, 12, 13, 14, 15, 16
 9,  8,  6, 12, 13, 14, 15, 16
 6,  8,  9, 12, 13, 14, 15, 16
 8,  6,  9, 12, 13, 14, 15, 16
 6,  8,  9, 12, 13, 14, 15, 16
```