JMBAG **IME I PREZIME**

Algoritmi i strukture podataka – 2. Ispitni rok

3. rujna 2013.

*Nije dopušteno korištenje globalnih i statičkih varijabli te naredbe **goto**. Neefikasna rješenja mogu donijeti manje bodova. Nerekurzivne funkcije se ne priznaju kao rješenja u zadatcima u kojima se traži rekurzivna funkcija. Pismeni ispit donosi najviše **70** bodova, s time da je za prolazak potrebno ostvariti najmanje **35** bodova. Četvrti zadatak rješavate na ovom obrascu, dok ostale zadatke rješavate na dodatnim papirima. Ovaj obrazac morate predati zajedno s cijelim Vašim uradkom.*

Zadatak 1. (13 bodova)

Za tip podatka **Stog** koji je ostvaren jednostruko povezanom listom definirane su funkcije za inicijalizaciju stoga, dodavanje jednog elementa na stog te uklanjanje jednog elementa sa stoga. Prototipovi navedenih funkcija su:

```
void init_stog(Stog *stog);
int dodaj(double element, Stog *stog);
int skini(double *element, Stog *stog);
```

Funkcije **dodaj** i **skini** vraćaju 1 ako je operacija dodavanja ili skidanja uspjela, a 0 inače. Napišite funkciju čiji je prototip:

```
int preurediStog(Stog *stog);
```

koja će promijeniti poredak elemenata u zadanom početnom stogu tako da najveći element stavi na vrh stoga, dok se poredak ostalih elemenata ne mijenja. Ako ima više jednakih, a najvećih, elemenata na vrh ih treba staviti sve, ali njihov poredak nije važan. Funkcija mora vratiti broj najvećih elementa.

Primjer: Stog s elementima (od vrha prema dnu) **3.1, 2.7, 4.5, 4.8, 4.5, 3.1, 4.8, 4.1** funkcija će promijeniti u **4.8, 4.8, 3.1, 2.7, 4.5, 4.5, 3.1, 4.1.**

Zadatak 2. (13 bodova)

Za tip podatka Red definirane su funkcije (nije ih potrebno pisati) za inicijalizaciju reda, dodavanje elementa u red te za skidanje elementa iz reda. Prototipovi navedenih funkcija su:

```
void init_red(Red *red);
int dodaj(int element, Red *red);
int skini(int *element, Red *red);
```

Funkcije **dodaj** i **skini** vraćaju 1 ako je operacija dodavanja ili skidanja uspjela, a 0 inače. Elementi reda su podatci tipa **int.** Tip **Red** definiran je sljedećim programskim odsječkom:

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

typedef struct {
	atom *ulaz;
	atom *izlaz;
} Red;
```

Napišite funkciju koja će iz zadanog reda u novi red izdvojiti sve elemente koji su manji od prosjeka elemenata u redu. U ulaznom redu trebaju ostati elementi koji nisu izdvojeni. Funkcija pozivatelju vraća taj novi red, a prototip je:

```
Red* IzdvojiManje(Red *red)
```

Zadatak 3. (19 bodova)

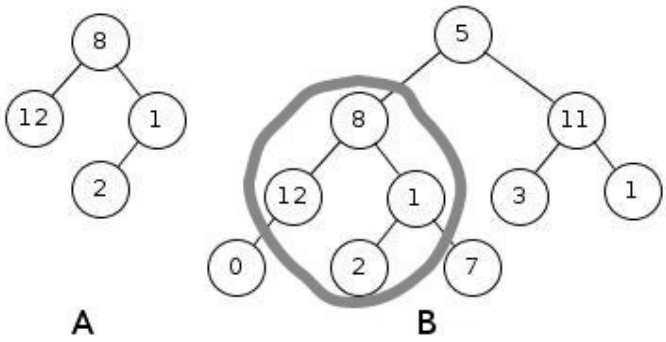
Binarno stablo sadrži cjele brojeve, a čvor je definiran sljedećom strukturom:

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

Napišite funkciju prototipa:

```
int jeLiDio(cvor * A, cvor* B, cvor *korijenA)
```

koja vraća 1 ako je stablo A ***dio*** stabla B, a 0 inače.



Slika 1 Primjer za zadatak

Stablo A je ***dio*** stabla B ako se unutar stabla B nalazi podstablo jednako stablu A, s time da stablo A koje je dio stabla B može imati i podstabla.

<u>Primjer:</u> Za stabla A i B na slici 1, stablo A je dio stabla B. Da npr. u čvoru 8 stabla B stavimo neku drugu vrijednost, stablo A više ne bi bilo dio stabla B.

Zadatak 4. (10 bodova)

Odredite asimptotsku složenost sljedećih algoritama:

a)
```
s=0;
for(i=1;i<=n;i*=2)
      for(j=1;j<=n;j++)
            s+=radi(n,j)+radi(n,i);
```

b)
```
s=0;
for(i=1;i<=n;i*=2)
      for(j=1;j<=n;j++)
            s+=radi(j,n)+radi(j,i);
```

uz pretpostavku da je asimptotska složenost funkcije radi(x,y) jednaka Θ(radi(x,y))=x. Obrazložite odgovor.

Zadatak 5. (15 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. Veličina bloka na disku je 2048 B. Očekuje se najviše 100000 zapisa, a kapacitet tablice je 25% veći. Prilikom upisa primjenjuje se metoda cikličkog preljeva. Ključ zapisa je *sifra*, a pretvorba ključa u adresu se obavlja već pripremljenom funkcijom

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

Napišite funkciju koja će dinamički alocirati jednodimenzionalno polje koje sadrži šifre onih zapisa koji su završili u preljevu i čija je cijena manja ili jednaka prosječnoj cijeni prozvoda. Funkcija treba vratiti pokazivač na alocirano polje i broj elemenata polja te treba imati prototip:

```
int* fun(FILE *f, int *brZapisa);
```

Rješenja:

Zad 1.

```
void init_stog(Stog *stog);
int dodaj(double element, Stog *stog);
int skini(double *element, Stog *stog);


int preuredi Stog(Stog *stog){
    Stog pom1, pom2;
    double element, max;
    int brPojavljivanja = 0;
    /*Inicijaliziraj stogove*/
    init_stog(&pom1);
    init_stog(&pom2);
    /*Trazimo najveci element - prvi element na stogu proglasavamo najvecim*/
    if(skini(&element, stog)){
        max = element;
        brPojavljivanja = 1; /*Broj povljivanja je 1*/
        dodaj(element, &pom1);
    }else{
        /*Stog je prazan*/
        return 0;
    }
    /*Provjera ostalih elemenata na stogu*/
    while(skini(&element, stog)){
        if(element > max){
            max = element;
            brPojavljivanja = 1; /*Novi najveci element - broj pojavljivanja je 1*/
        }else if(element == max){
            brPojavljivanja++;
        }
        dodaj(element, &pom1);
    }
    /*Prvo u stog zapisujemo elemente koji su manji od najveceg*/
    while(skini(&element, &pom1)){
        if(element == max){
            /*Sva pojavljivanja najveceg elementa spremamo na pom2 stog*/
            dodaj(element, &pom2);
            continue;
        }
        dodaj(element, stog);
    }
    /*Sada na vrh stoga dodajemo sva pojavljivanja najveceg elementa*/
    while(skini(&element, &pom2)){
        dodaj(element, stog);
    }
    return brPojavljivanja;
}
```

Zad 2.

```
Red *IzdvojiManje(Red *red){
          Red pom; Red *novi;
          int br=0,el; double suma=0;
          novi = (Red*)malloc(sizeof(Red));
          init_red(&pom);init_red(novi);
          while(SkiniIzReda(&el, red))
          {
                    suma+=el;
                    br++;
                    DodajURed(el, &pom);
          }
          suma=suma/br;
          while (SkiniIzReda(&el,&pom))
          {
                    if(el<suma) DodajURed(el,novi);
                    else DodajURed(el,red);
          }
          return novi;
}
```

Zad 3.

```
int jeLiDio(cvor* A, cvor* B, cvor* korijenA){
    int test;
    if (A == NULL)
        return 1;
    if (B == NULL)
        return 0;
    if (A->vrijednost == B->vrijednost){
        test=jeLiDio(A->lijevo, B->lijevo, korijenA) && jeLiDio(A->desno, B->desno, korijenA);
        if (test) return 1;
    }
    if (A == korijenA)
        return jeLiDio(korijenA, B->lijevo, korijenA) || jeLiDio(korijenA, B->desno, korijenA);
    return 0;
}
```

Zad4.

a) $2n^2\log_2 n$ b) $n(n+1)\log_2 n$

Zad5.

```
#define N 100000
#define BLOK 2048
#define C BLOK/sizeof (zapis)
#define M (int)(N*1.25/C)

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

int* fun(FILE *f, int *brZapisa){
  zapis pretinac[C];
  int i, j, predvidjeni_pretinac;
  int br_zapisa = 0;
  double suma = 0;
  double prosjek = 0;
  int *polje = NULL;

  //Brojanje zapisa
  for (i = 0; i < M; i++) {
    fseek (f, i*BLOK, SEEK_SET);
    fread (pretinac, sizeof (pretinac), 1, f);
    for (j = 0; j < C; j++) {
      if (pretinac[j].sifra != 0) { //preskačemo prazne zapise
        br_zapisa++;
        suma += pretinac[j].cijena;
      }
      else
        break; //nakon prvog praznog u pretincu, svi su ostali prazni
    }
  }

   //Racunanje prosjeka
   prosjek = (double) suma / br_zapisa;

   //Brojimo samo zapise koji su u preljevu i kojima je cijena <= prosjek
   br_zapisa = 0;

   //Trazenje elemenata i punjenje polja
  for (i = 0; i < M; i++) {
    fseek (f, i*BLOK, SEEK_SET);
    fread (pretinac, sizeof (pretinac), 1, f);
    for (j = 0; j < C; j++) {
      if (pretinac[j].sifra != 0) { //preskačemo prazne zapise
        predvidjeni_pretinac = adresa(pretinac[j].sifra);
        if (predvidjeni_pretinac != i && pretinac[j].cijena <= prosjek) {
          polje = (int*) realloc(polje, (++br_zapisa) * sizeof(int));
          polje[br_zapisa-1] = pretinac[j].sifra;
        }
      }
      else
        break; //nakon prvog praznog u pretincu, svi su ostali prazni
    }
 }

  *brZapisa = br_zapisa;

  return polje;
}
```