JMBAG **IME I PREZIME**

Algoritmi i strukture podataka – Završni ispit

16. lipnja 2015.

*Nije dopušteno korištenje globalnih i statičkih varijabli te naredbe **goto**. Neučinkovita rješenja mogu nositi manje bodova. Ispit donosi maksimalno 35 bodova, a prag za prolaz na ispitu je **10** bodova.*

2. zadatak rješavate na ovom obrascu, a ostale zadatke na svojim listovima papira. Ovaj obrazac trebate predati s upisanim identifikacijskim podacima na vrhu obrasca.

Zadatak 1. (8 bodova)

Za tip podatka `Stog` definirane su funkcije za inicijalizaciju stoga, dodavanje elementa na stog te skidanje elementa sa stoga. Elementi stoga su realni brojevi. 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. Potrebno je napisati funkciju `otpadnici` čiji je prototip:

```
Stog otpadnici(Stog *S, double d);
```

Funkcija `otpadnici` iz stoga `S` uklanja sve elemente koji su od aritmetičke sredine elemenata stoga `S` udaljeni barem za d > 0 (ne treba provjeravati je li d > 0). Uklonjeni elementi se stavljaju na stog `T` i to tako da su prvo, ako postoje, stavljeni oni elementi koji su manji od aritmetičke sredine (poredak nije bitan), a onda, ako postoje, oni elementi koji su veći od aritmetičke sredine (poredak nije bitan). Na stogu `S`, poredak elemenata koji nisu uklonjeni treba ostati isti.

Funkcija treba biti neovisna o implementaciji tipa podataka `Stog`, a dozvoljeno je korištenje pomoćnih stogova.

Primjer:

Zadan je stog `S`: (dno stoga) `4 < 5 < 6 < 4 < 1` (vrh stoga). Aritmetička sredina elemenata na stogu je 4.

Ako je funkcija pozvana za d = 2, na izlazu iz funkcije stog `S` treba biti `4 < 5 < 4`, a stog `T` `1 < 6`. Ako je funkcija pozvana za d = 4, na izlazu iz funkcije, stog `S` se ne mijenja, a stog `T` je prazan.

Zadatak 2. (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/=2) {
    for (j=0;j<i;j++) {
      printf("%d\n",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 računa faktorijel argumenta, a modulo operator u argumentu printf-a osigurava jednako vrijeme izvođenja printf-a bez obzira na veličinu argumenta)

Zadatak 3. (6 bodova)

U jednostruko povezanu nesortiranu listu spremljeni su podaci studentima. Lista je zadana sljedećim strukturama:

```
struct zapis {
  char imeprezime[80+1];
  char spol;
  int dob;
};
```

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

Napišite funkciju prototipa:

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

koja će elemente liste (zapise o studentima) rasporediti ovisno o spolu. U novoj listi, zapisi o osobama ženskog spola (spol 'Z') trebaju se nalaziti na početku liste, a zapisi o osobama muškog spola (spol 'M') na kraju. Funkcija ne smije alocirati nove čvorove.

Primjer:

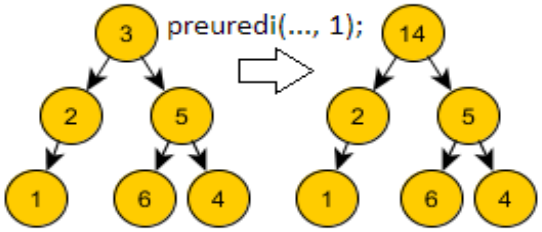
Listu (od početka prema kraju): `("ivo",'M',19);("ana",'Z',18);("mara",'Z',18);("mate",'M',19);("sasa",'Z',21);`

funkcija transformira u: `("ana",'Z',18);("mara",'Z',17);("sasa",'Z',21);("ivo",'M',19);("mate",'M',19);`

Zadatak 4. (6 bodova)

Čvor stabla je zadan je strukturom:

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



a) Napisati rekurzivnu funkciju, koja će za svaki čvor stabla na razinama čiji je paritet `paritet`, izračunati novu vrijednost kao zbroj inicijalne vrijednosti tog čvora i vrijednosti svih njegovih potomaka na razini `razina+2`. Prototip funkcije jest:

```
int preuredi (cvor* korijen, int razina, int paritet);
```

(napomena: argumentom paritet iz skupa {1,0} se funkciji zadaje uvećava li vrijednosti na neparnim ili na parnim razinama stabla; npr. ako je paritet 1, onda za zadanu razina koja je neparan broj, treba uvećati razinu korijena te svaku drugu od nje, a ako je paritet 0,onda ostale. Crtež pokazuje primjer transformacije za poziv s argumentom 1.)

b) Napisati dio koda koji postiže da je cijelo stablo promijenjeno na način naveden u a) dijelu zadatka.

Zadatak 5. (4 boda)

Prioritetni red `P` realiziran je gomilom s relacijom manji od (*min. heap*). Gomila je implementirana poljem cjelobrojnih elemenata. Polje može imati najviše 15 članova, a sadržaj polja na početku izvođenja je:

6	10	9	11	20	25	19	24	16						

Funkcija zadana prototipom:

```
void DodajURed( PRed *P, int e);
```

realizira ubacivanje elementa u gomilu algoritmom čije je vrijeme izvođenja $O(\log_2 n)$.

Funkcija zadana prototipom:

```
void SkiniIzReda(PRed *P, int *e);
```

vraća vršni element iz gomile te nakon toga podešava gomilu algoritmom čije je vrijeme izvođenja $O(\log_2 n)$.

Ilustrirati sadržaj prioritetnog reda P nakon svake zamjene dvaju elemenata, ako su dodavani i skidani elementi sljedećim nizom naredbi (za gore zadani prioritetni red P):

```
DodajURed(&P, 5);
SkiniIzReda(&P, &e);
SkiniIzReda(&P, &e);
DodajURed(&P, 21);
```

Rješenja:

Zadatak 1. (8 bodova)

```
Stog otpadnici(Stog *stog, double d){
	int stavka, suma;
	double prosjek;
	int broj;
	Stog pomStog, stogManjih,  stogVecih;

	init_stog(&pomStog);
	if(!skini(&stavka, stog)) {
		return pomStog;
	}
	dodaj(stavka,&pomStog);

	suma=stavka;
	broj=1;
	/* dok je elemenata na stogu */
	while(skini(&stavka,stog)){
	    /* računaj sumu svih */
	    suma+=stavka;
	    broj++;
	    /* i stavi elemente na pomoćni stog (obrnuti poredak): */
	    dodaj(stavka,&pomStog);
	}
	prosjek=(double)(suma)/broj;
	init_stog(&stogManjih);
	init_stog(&stogVecih);
	/* vraćam sa pomStog na stog, a one koji su udaljeni bacam u pomoćne stogove */
	while(skini(&stavka,&pomStog)){
		if (stavka <= prosjek-d){
			dodaj(stavka,&stogManjih);
		}
		else if(stavka>= prosjek+d){
			dodaj(stavka,&stogVecih);
		}
		else dodaj(stavka,stog);
	} /* stog je u pravom poretku, a stogManjih i stogVecih su u obrnutom poretku,
	  ali to nije bitno */

	/* prepiši stogManjih na pomStog */
	while(skini(&stavka,&stogManjih)){
		dodaj(stavka,&pomStog);
	}
	/* prepiši stogVecih na pomStog */
	while(skini(&stavka,&stogVecih)){
		dodaj(stavka,&pomStog);
	}
	return pomStog;
}
```

Zadatak 2. (6 bodova)

O(n) (**2B**) (jer (N+N/2+N/4+...+1) = 2N, pa je i dolje omega od N)	O(n^2) (**2B**) (poziv fact ne ulazi u računicu za velike n-ove, pa tako ni u O notaciju)
Ω(n) (**1B**)	Ω(n) (**1B**)

Zadatak 3. (6 bodova)

```
void obradiListu(atom **glava) {
	atom *prvi = NULL, *prva = NULL,
		*zadnji = NULL, *zadnja, *trenutni,
		*sljedeci;

	trenutni = (*glava);

	while (trenutni) {
		sljedeci = trenutni->sljed;
		if (trenutni->element.spol == 'Z') {
			if (!prva) {
				prva = zadnja = trenutni;
			}
			else {
				zadnja = zadnja->sljed = trenutni;
			}
		}
		else {
			if (!prvi) {
				prvi = zadnji = trenutni;
			}
			else {
				zadnji = zadnji->sljed = trenutni;
			}
		}
		trenutni = sljedeci;
	}
	if (zadnji) {
		zadnji->sljed = NULL;
	}
	if (prva) {
		zadnja->sljed = prvi;
		*glava = prva;
	}
	else
		*glava = prvi;
}
```

Zadatak 4. (6 bodova)

a)

```
int preuredi (cvor* korijen, int razina, int paritet)
{
  if (cvor==NULL) return 0;
  int resL=preuredi(korijen->lijevo,razina+1,parity);
  int resR=preuredi(korijen->desno,razina+1,parity);
  if (razina%2==paritet)
  {
    int old=korijen->vrijednost

    korijen->vrijednost+=resL+resR;
    return old
  }
  return resL+resR;
}
```

b)

```
/* korijen stabla je pokazivac korijen*/
cvor * korijen;
…
preuredi (korijen,1,0);
preuredi (korijen,1,1);
```

Zadatak 5. (4 bodova)

(različite razine stabla označene su različitim bojama za lakše snalaženje, elementi koji idu u zamjenu ili brisanje u idućem koraku su potcrtani)

6	10	9	11	20	25	19	24	16						
6	10	9	11	20	25	19	24	16	5					
6	10	9	11	5	25	19	24	16	20					
6	5	9	11	10	25	19	24	16	20					
5	6	9	11	10	25	19	24	16	20					
20	6	9	11	10	25	19	24	16						
6	20	9	11	10	25	19	24	16						
6	10	9	11	20	25	19	24	16						
16	10	9	11	20	25	19	24							
9	10	16	11	20	25	19	24							
9	10	16	11	20	25	19	24	21						