JMBAG **IME I PREZIME**

Algoritmi i strukture podataka – jesenski ispitni rok

6. rujna 2016.

Nije dopušteno korištenje globalnih i statičkih varijabli te naredbe goto. Ovaj primjerak ispita trebate predati s upisanim imenom i prezimenom te JMBAG-om.
*Ispit donosi maksimalno 70 bodova, a prag za prolaz pismenog ispita je **35** bodova uz barem jedan točno riješen zadatak.*

Zadatak 1. (20 bodova)

a) Potrebno je definirati strukturu **zOsoba** koja će sadržavati šifru osobe (cijeli broj), ime osobe (30 + 1 znakova) i prezime osobe (40 + 1 znakova).

b) Binarno stablo sadrži zapise o osobama (koristi se struktura **zOsoba**), a čvorovi su definirani strukturom **cvorBS**:

```
typedef struct cvBS {
  zOsoba osoba;
  struct cvBS *lijevo, *desno;
} cvorBS;
```

Čvorovi u stablu uređeni su prema šifri osobe tako da lijevo dijete sadrži šifru koja je manja od ili jednaka šifri u čvoru-roditelju, a desno dijete sadrži šifru osobe koja je veća od šifre u čvoru-roditelju.

Potrebno je napisati funkciju **stvoriListu** koja će stvoriti jednostruko povezanu listu koja će sadržavati sve zapise o osobama iz binarnog stabla tako da zapisi u listi budu uzlazno poredani prema šifri osobe. Čvorovi u jednostruko povezanoj listi definirani su strukturom **cvorL**:

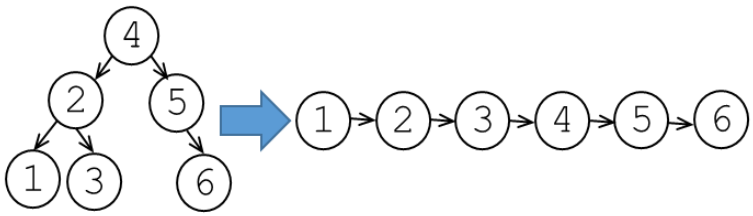
```
typedef struct cvL {
  zOsoba osoba;
  struct cvL *sljed;
} cvorL;
```

Prototip funkcije je:

```
int stvoriListu(cvorBS *korijen, cvorL **glava, cvorL **zadnji);
```

Po izlasku iz funkcije pokazivač **glava** treba pokazivati na početni čvor u listi (koji sadrži najmanju šifru), a pokazivač **zadnji** na zadnji čvor u listi. Funkcija vraća vrijednost 1, ako su svi zapisi o osobama uspješno prepisani u listu, a 0 inače.

Na slici je prikazan primjer binarnog stabla te liste koja se dobije prepisivanjem zapisa o osobama korištenjem funkcije **stvoriListu**. Zbog jednostavnosti, u čvorovima stabla i liste prikazane su samo šifre osoba.



c) Pretpostavite da binarno stablo ima **n** čvorova. Odredite vrijeme izvođenja funkcije **stvoriListu** iz b) dijela zadatka u O, Ω i Θ notaciji:

O _____ Ω _____ Θ_____

Zadatak 2. (10 bodova)

U cjelobrojnom polju pohranjen je sljedeći niz brojeva:

5, 12, 2, 18, 3, 9, 4, 1, 16, 13

Ilustrirajte uzlazno sortiranje algoritmom *quicksort*. Stožer se odabire metodom aproksimacije medijana temeljem prvog, srednjeg i zadnjeg člana. Potpolje koje ima najviše 3 člana sortira se izravno bez navođenja koraka.
Potrebno je prikazati sadržaj polja nakon svake promjene.

Zadatak 3. (30 bodova)

a) Potrebno je napisati rekurzivnu funkciju **hanoiStog**, koja sve poteze u igri Hanojski tornjevi pohranjuje na stog (na vrhu stoga je uvijek potez koji je odigran posljednji):

```
void hanoiStog (char izvor, char pomocni, char odrediste, int n, Stog *stog);
```

Argumenti **izvor**, **pomocni** i **odrediste** predstavljaju izvorni, pomoćni i odredišni toranj (oznake 'I', 'O' ili 'P'). Kod inicijalnog poziva funkcije stog je prazan, a argument **n** predstavlja broj diskova na tornju **izvor** (**n** ≥ 1).

Definirana je struktura **potez** i pomoćne funkcije za rad sa stogom:

```
typedef struct s {
  char izvor, odrediste; /* disk se prebacuje s izvora na odrediste */
  int disk; /* redni broj diska koji se prebacuje; najmanji disk ima redni broj 1 */
} potez;

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

Napomena: nerekurzivno rješenje se neće priznati.

b) Potrebno je napisati funkciju **ispisPoteza** koja ispisuje sve poteze u igri Hanojski tornjevi redoslijedom kojim su odigrani (prvi potez treba biti ispisan prvi). Funkcija se poziva s argumentom **stog**, čiji vrh, u trenutku poziva funkcije, pokazuje na potez koji je odigran posljednji. Prototip funkcije je:

```
void ispisPoteza (Stog *stog);
```

Primjer: za n = 2 funkcija `ispisPoteza` treba ispisati:

```
Prebacujem element 1 s tornja I na toranj P
Prebacujem element 2 s tornja I na toranj O
Prebacujem element 1 s tornja P na toranj O
```

c) Odredite vremena izvođenja u O, Ω i Θ notaciji za funkcije **hanoiStog** i **ispisPoteza** iz a) i b) dijela zadatka (u ovisnosti o broju diskova **n**):

Funkcija **hanoiStog**: O _____ Ω _____ Θ_____

Funkcija **ispisPoteza**: O _____ Ω _____ Θ_____

Zadatak 4. (10 bodova)

Za zadani program prikažite sadržaj sistemskog stoga počevši s pozivom funkcije **noc** u glavnome programu (linija 19) te sve do neposredno prije izvođenja naredbe **return 0;** u funkciji **dan** (linija 9). Uz svaku stavku napišite broj okteta (*byte*-ova) koje stavka zauzima na stogu.

Sadržaj stoga te broj okteta koje zauzima svaka stavka prikažite u pravokutniku desno od zadanog programa.

```
1  #include <stdio.h>
2  void noc(int *a, int *b);
3  int dan(int *a, int *b) {
4    if (*a < *b) {
5      *a += 3;
6      noc(a, b);
7    }
8    else {
9      return 0;
10   }
11 }
12
13 void noc(int *a, int *b) {
14   dan(a, b);
15 }
16
17 int main(void) {
18   int A[] = { 5, 6, 7, 8 };
19   noc(&A[0], &A[1] + 2);
20   return 0;
21 }
```

Sadržaj stoga:

Rješenja:

1. zadatak (20 bodova)

a)
```
typedef struct o {
  int sifOsoba;
  char imeOsoba[30 + 1];
  char prezOsoba[40 + 1];
} zOsoba;
```

b)
```
int stvoriListu(cvorBS *korijen, cvorL **glava, cvorL **zadnji) {
  cvorL *p = NULL;
  if (korijen != NULL) { // inorder obilazak stabla
    stvoriListu(korijen->lijevo, glava, zadnji);
    p = (cvorL *)malloc(sizeof(cvorL));
    if (p) {
      p->osoba = korijen->osoba;
      p->sljed = NULL;
      if (*glava == NULL) { // prvi cvor u listi
        *glava = p;
      }
      else {
        (*zadnji)->sljed = p;
      }
      *zadnji = p;
    }
    else {
      return 0;
    }
    stvoriListu(korijen->desno, glava, zadnji);
  }
  return 1;
}
```

c) O(n), Ω(n), Θ(n)

2. zadatak (10 bodova)

5, 12, 2, 18, **3**, 9, 4, 1, 16, **13**	odabir stožera
3, 12, 2, 18, **5**, 9, 4, 1, 16, **13**	
3, 12, 2, 18, **16**, 9, 4, 1, **5**, 13	sklanjanje stožera
3, **1**, 2, 18, 16, 9, 4, **12**, 5, 13	zamjena 1 i 12
3, 1, 2, **4**, 16, 9, **18**, 12, 5, 13	zamjena 4 i 18
3, 1, 2, 4, **5**, 9, 18, 12, **16**, 13	obnova stožera
1, 3, 2, 4, 5, 9, 18, 12, 16, 13	poredak nakon odabir stožera u LP
1, 2, **3**, 4, 5, 9, 18, 12, 16, 13	skloni stožer u LP; nema zamjena i nova potpolja se izravno sortiraju
1, 2, 3, 4, 5, **9, 18, 12**, 16, **13**	odabir stožera u DP; nema zamjena u poretku
1, 2, 3, 4, 5, 9, 18, 16, **12**, 13	skloni stožer
1, 2, 3, 4, 5, 9, **12**, 16, **18**, 13	obnovi stožer
1, 2, 3, 4, 5, 9, 12, 16, 18, 13	potpolja se izravno sortiraju
1, 2, 3, 4, 5, 9, 12, **13**, 18, **16**	zamjena 13 i 16
1, 2, 3, 4, 5, 9, 12, 13, **16**, **18**	zamjena 18 i 16
1, 2, 3, 4, 5, 9, 12, 13, 16, 18	

3. zadatak (30 bodova)

a)

```
void hanoiStog(char izvor, char odrediste, char pomocni, int n, Stog *stog) {
  potez novi;
  if (n > 0) {
    hanoiStog(izvor, pomocni, odrediste, n - 1, stog);
    novi.izvor = izvor;
    novi.odrediste = odrediste;
    novi.disk = n;
    dodaj(stog, novi);
    hanoiStog(pomocni, odrediste, izvor, n - 1, stog);
  }
}
```

b)

```
void ispisPoteza(Stog *stog) {
  potez element;
  Stog pomStog;
  init_stog(&pomStog);
  while (skini(stog, &element)) {
    dodaj(&pomStog, element);
  }
  while (skini(&pomStog, &element)) {
    printf("Prebacujem element %d s tornja %c na toranj %c\n", element.disk, element.izvor,
element.odrediste);
  }
}
```

c)

Funkcija **hanoiStog**: $O(2^n)$, $\Omega(2^n)$, $\Theta(2^n)$

Funkcija **ispisPoteza**: $O(2^n)$, $\Omega(2^n)$, $\Theta(2^n)$

4. zadatak (10 bodova)

Poziv dan	Povr. adresa dan	4
	a	4
	b	4
Poziv noc	Povr. adresa noc	4
	a	4
	b	4
Poziv dan	Povr. adresa dan	4
	a	4
	b	4
Poziv noc	Povr. adresa noc	4
	&A[0]	4
	&A[1] + 2	4