Algoritmi i strukture podataka – Završni ispit

20. lipnja 2012.

Nije dopušteno korištenje globalnih i statičkih varijabli te naredbe ***goto****.*

Zadatak 1. (6 bodova)

Za red realiziran jednostruko povezanom listom napišite funkciju **dodaj** za dodavanje elementa u red. Prototip funkcije je:

```
int dodaj(int element, atom **ulaz, atom **izlaz);
```

Funkcija treba vratiti 1 ukoliko je element uspješno dodan u red, a 0 inače.
Tip `atom` definiran je sljedećim programskim odsječkom:

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

Zadatak 2. (12 bodova)

Zadan je niz brojeva 2, 5, 7, 8, 11, 1, 4, 2, 3, 7 koji su spremljeni u čvorove binarnog stabla. Čvor stabla definiran je sljedećim programskim odsječkom:

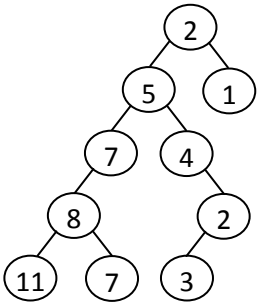
```
typedef struct cv {
  int el;
  struct cv *lijevo;
  struct cv *desno;
} cvor;
```

a)*(6 bodova)* Napišite funkciju **dodaj** koja dodaje element u stablo tako da se pozivima te funkcije za sve elemente zadanog niza brojeva stvori stablo kao na slici Slika 1. Funkcija vraća pokazivač na korijen stabla. Prototip funkcije je:

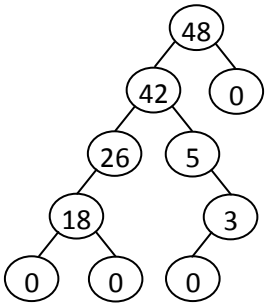
```
cvor *dodaj (cvor *korijen, int broj);
```

b)*(3 boda)* Napišite funkciju **zamijeni** koja će svaki element stabla zamijeniti sumom elemenata u njegovom lijevom i desnom podstablu (tj. sumom njegovih potomaka **prije zamjene** vrijednosti u tim potomcima). Npr. stablo sa slike Slika 1 transformirat će se u stablo na slici Slika 2.

c)*(3 boda)* Napišite glavni program koji će definirati cjelobrojno polje i inicijalizirati ga na vrijednosti iz zadanog niza brojeva. Korištenjem funkcije iz a) dijela zadatka stvoriti stablo sa slike Slika 1, te pozivom funkcije iz b) dijela zadatka transformirati ga u stablo na slici Slika 2.



Slika 1

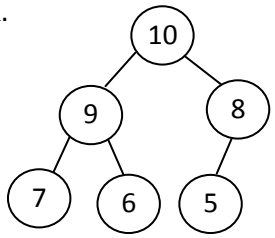


Slika 2

Zadatak 3. (4 boda)

a)*(2 boda)* Zadan je **inorder** (lijevo, korijen, desno) ispis gomile u kojoj je roditelj veći od svoje djece: 75, 78, 74, 80, 11, 32, 92, 50, 84, 58. Nacrtajte gomilu.

b)*(2 boda)* Zadana je gomila kao na slici. Prikažite sve korake uzlaznog heap sorta



Zadatak 4. (8 bodova)

Za tip podatka `Stog` koji je realiziran jednostruko povezanom listom definirane su funkcije za inicijalizaciju stoga, dodavanje elementa na stog te skidanje elementa sa stoga. Zadana su dva stoga `stog1` i `stog2`, a elementi stogova su podaci tipa `Osoba` koji sadrže informaciju o šifri osobe (cijeli broj) i visini (cijeli broj). Stog `stog1` sadrži podatke samo za muškarce, a `stog2` samo za žene i na oba stoga su podaci sortirani po visini tako da je najviša osoba na vrhu stoga. Prototipovi navedenih funkcija su:

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

Funkcije `dodaj` i `skini` vraćaju 1 ako je operacija dodavanja ili skidanja uspjela, a 0 inače.

a)*(1 bod)* Napišite **definicije** odgovarajućih struktura (tipovi podataka **`Osoba`**, **`Stog`** i **`atom`**) za stog realiziran jednostruko povezanom listom.

b)*(7 bodova)* Napišite **rekurzivnu** funkciju **`spoji`** koja će sa stoga `stog2` skinuti sve podatke i staviti ih na stog `stog1`, ali tako da ostane očuvan poredak po visini. Poredak osoba s istom visinom nije bitan.

Primjer:

stog1	**stog2**	**stog1 (nakon spoji)**
vrh stoga-> 345, 202	vrh stoga-> 348, 203	vrh stoga-> 348, 203
161, 200	234, 200	345, 202
654, 197	111, 189	234, 200
122, 182	981, 169	161, 200
234, 170		654, 197
		111, 189
		122, 182
		234, 170
		981, 169

Zadatak 1. (6 bodova)

```
int dodaj(int element, atom **ulaz, atom **izlaz){
  atom *novi;
  if (novi = malloc (sizeof (atom))) {
    novi->element = element;
    novi->sljed = NULL;
    if (*izlaz == NULL)
      *izlaz = novi;  // ako je red bio prazan
    else
      (*ulaz)->sljed = novi; // inace, stavi na kraj
    *ulaz = novi;        // zapamti zadnjeg
    return 1;
  }
  return 0;
}
```

Zadatak 2. (12 bodova)

a) (6 bodova)

```
cvor *upis (cvor *korijen, int broj) {
  if (korijen == NULL) {
    korijen = (cvor *) malloc (sizeof (cvor));
    if (korijen) {
        korijen->element = broj;
        korijen->lijevo = korijen->desno = NULL;
    }
    else
      return NULL;

  }
  else if (broj >= korijen->element)
    korijen->lijevo = upis (korijen->lijevo, broj);
  else
    korijen->desno  = upis (korijen->desno, broj);
  return korijen;
 }
```

b)(3 boda)

```
int zamijeni (cvor *korijen){
  int el;
  if(!korijen) return 0;
  el = korijen->el;
  korijen->el = zamijeni (korijen->lijevo)+zamijeni (korijen->desno);
  return el+korijen->el;
}
```

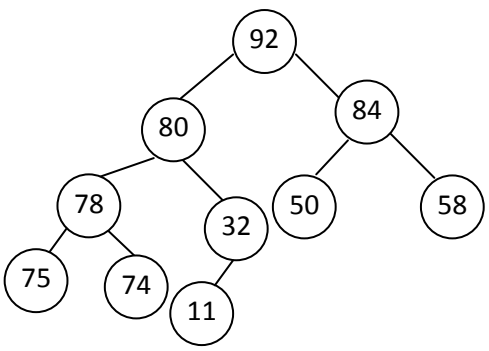
c)(3 boda)

```
int main() {
  int i;
  cvor *korijen=NULL;
  int polje [10]={2,5,7,8,11,1,4,2,3,7};
  for (i=0; i<10;i++)
    korijen = upis (korijen, polje[i]);
  zamijeni(korijen);
  return 0;
}
```

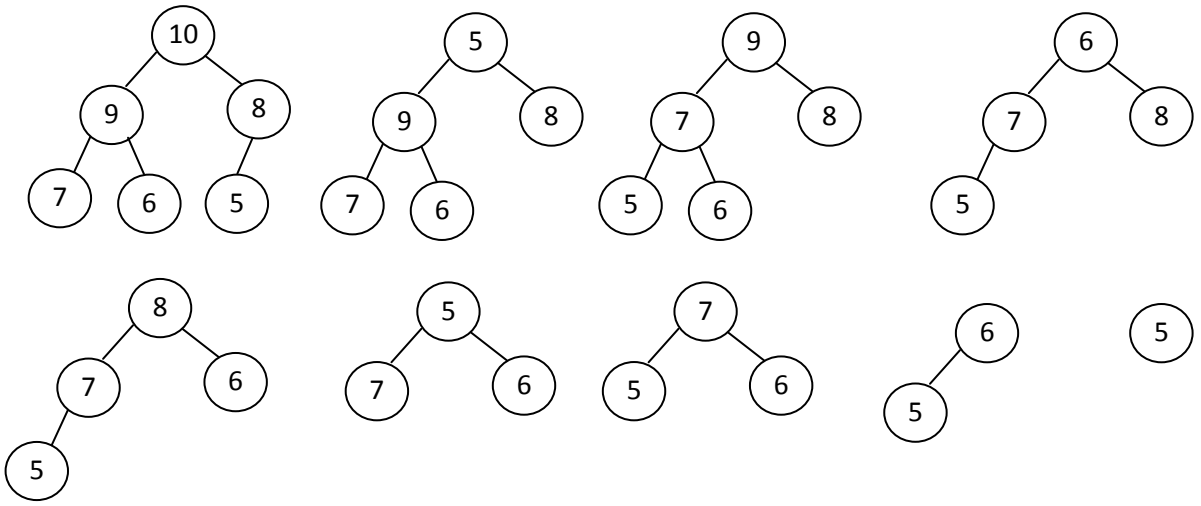
Zadatak 3. (2 + 2 boda)

75, 78, 74, 80, 11, 32, 92, 50, 84, 58.

a)



b)



Zadatak 4. (8 bodova)

a)(1bod)

```
typedef struct {

       int sifra;
       int visina;
} Osoba;

struct at {
  Osoba element;
  struct at *sljed;
};
typedef struct at atom;
typedef struct{
       atom *vrh;
} Stog;
```

b) (7 bodova)

```
void spoji (Stog *s1, Stog *s2){
  Osoba os1, os2, pom;
  int imas1, imas2;
  imas1=skini(&os1, s1);
  imas2=skini(&os2, s2);
  if (!imas1 && !imas2) return;
  else if (imas1 && imas2){
    if (os1.visina <os2.visina)  {
      dodaj (os1,s1);
      pom=os2;
    }
    else{
      dodaj (os2,s2);
      pom=os1;
    }
  }
  else if (imas1)
    pom=os1;
  else
    pom=os2;

  spoji (s1,s2);
  dodaj (pom,s1);
}
```