JMBAG **IME I PREZIME**

Algoritmi i strukture podataka – završni ispit

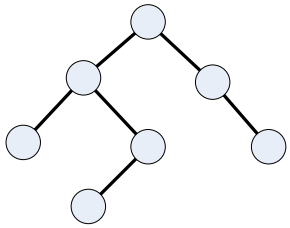
13. lipnja 2017.

Nije dopušteno korištenje globalnih i statičkih varijabli te naredbe goto. Ispit donosi maksimalno 40 bodova. Ovaj primjerak ispita trebate predati s upisanim imenom i prezimenom te JMBAG-om.

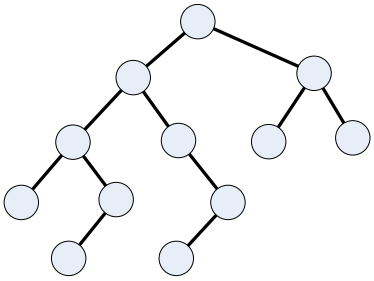
Zadatak 1. (10 bodova)

Potrebno je provjeriti je li binarno stablo balansirano s obzirom na visinu. Prazno stablo je balansirano, a neprazno binarno stablo T je balansirano ako vrijedi:

1) Lijevo i desno podstablo od T je balansirano
2) Razlika u visini lijevog i desnog podstabla nije više od 1



Stablo balansirano po visini



Stablo koje nije balansirano po visini

a) Napisati funkciju koja će izračunati visinu stabla. Ako je stablo prazno vraća 0. Prototip funkcije:

```
int visina(cvor *korijen);
```

b) Napisati funkciju koja će provjeriti da li je stablo balansirano. Ako je balansirano vraća 1, inače 0. Dopušteno je koristiti funkciju `visina`. Prototip funkcije:

```
int balans(cvor *korijen);
```

Čvorovi stabla imaju sljedeću strukturu:

```
typedef struct cv{
      int podatak;
      struct cv *lijevo_dijete;
      struct cv *desno_dijete;
} cvor;
```

Zadatak 2. (10 bodova)

Zadan je niz brojeva: **12, 48, 54, 22, 81, 62, 71, 32.**

a) **(5 bodova)** Ilustrirajte (nacrtajte stablo nakon svake promjene) stvaranje gomile (max heap) od zadanog polja brojeva algoritmom čija je složenost za najgori slučaj *O(n).*

b) **(5 bodova)** Ilustrirajte (nacrtajte stablo nakon svake promjene) stvaranje gomile (max heap) od zadanog polja brojeva algoritmom čija je složenost za najgori slučaj *O(nlogn).*

Zadatak 3. (10 bodova)

Podatke o studentima potrebno je organizirati u memorijski rezidentnu tablicu raspršenog adresiranja veličine 10000 zapisa. Memorija za tablicu raspršenog adresiranja rezervira se u glavnom programu (nije ga potrebno pisati). Kolizija se rješava linearnim ispitivanjem.

Potrebno je napisati:

a) Pretprocesorsku direktivu kojom se definira veličina tablice te ostali potrebni parametri u strukturi. Potrebno je definirati strukturu `zapis` za pohranu podataka o studentu. Svaki student ima šifru (cijeli broj), ime (maksimalna veličina 20 znakova) i prezime (maksimalna veličina 30 znakova).
b) Funkciju koja će odrediti na koju adresu je bilo upućeno najviše zapisa. Funkcija treba vratiti tu adresu i broj zapisa upućenih na nju (lokacija u tablici), a koji su završili u preljevu. Šifra nula (`0`) označava prazan zapis. Ako ima više takvih adresa vratiti rezultat za bilo koju od njih.
```
int Provjera(zapis* hash, int *brojPreljeva);
```

Prilikom realizacije funkcije za upis koristiti predefiniranu hash-funkciju:

```
int Adresa(int sifra);
```

Zadatak 4. (10 bodova)

Zadan je tip podatka **`Red`** za koji su definirane funkcije 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 uspjela, a 0 ako nije.

Napišite funkciju koja će sva pojavljivanja višekratnika zadanog broja iz zadanog reda izdvojiti u novi red koji treba vratiti pozivatelju. Prototip funkcije je:

```
Red *izdvojiVisekratnike(Red *red, int broj);
```

U ulaznom redu moraju ostati svi preostali elementi.

Napišite i dio glavnog programa u kojem ćete definirati redove i ispravno pozvati napisanu funkciju. Dio koda u kojem se puni red nije potrebno pisati.

Primjer: Za ulazni red sadržaja **3, 8, 2, 5, 35, 2, 24, 15** poziv funkcije **izdvojiVisekratnike** za **broj=3** stvara novi red sadržaja **3, 24, 15** dok u starom redu ostaju **8, 2, 5, 35, 2**.

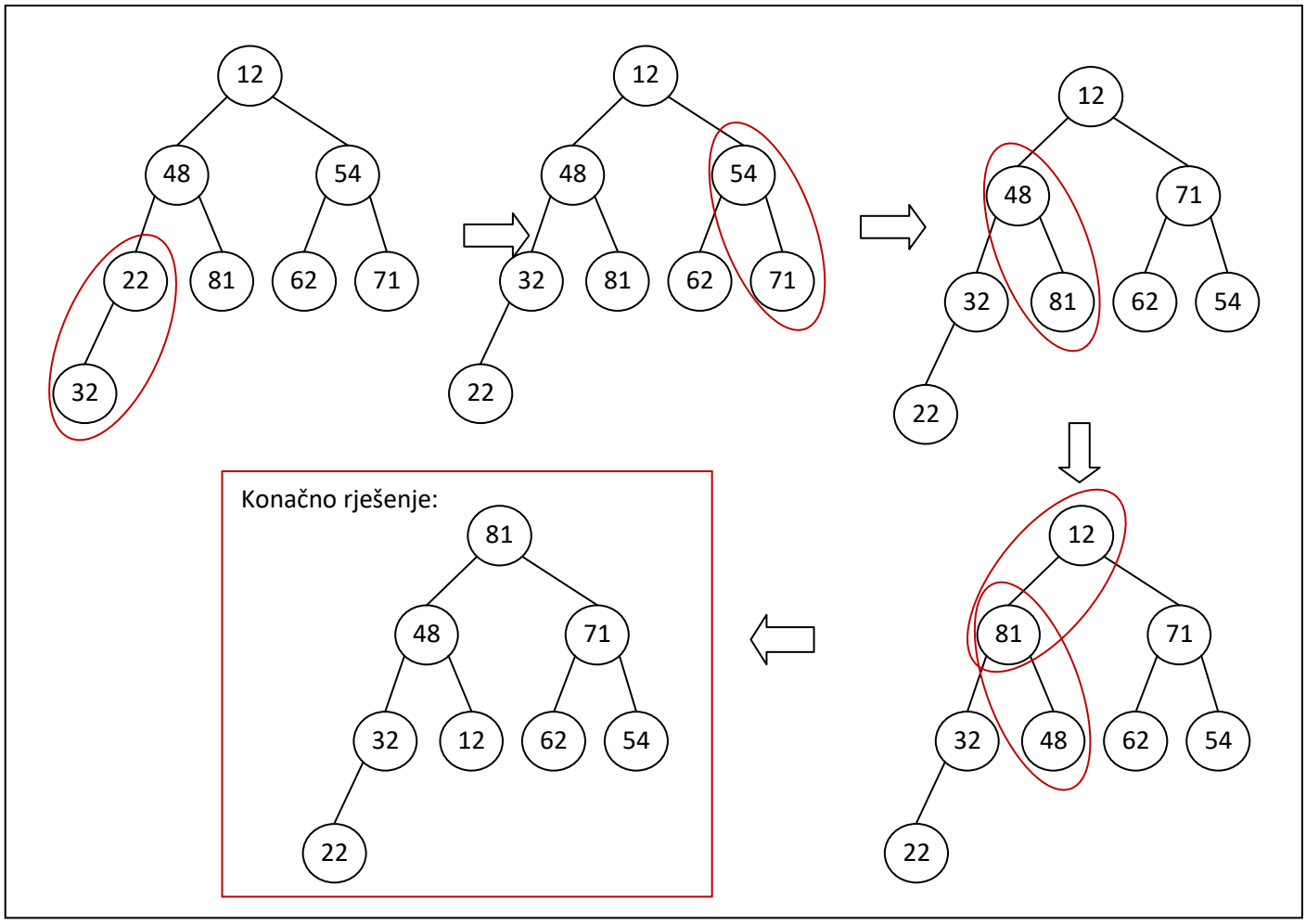
Rješenja:

1. zadatak (10 bodova)

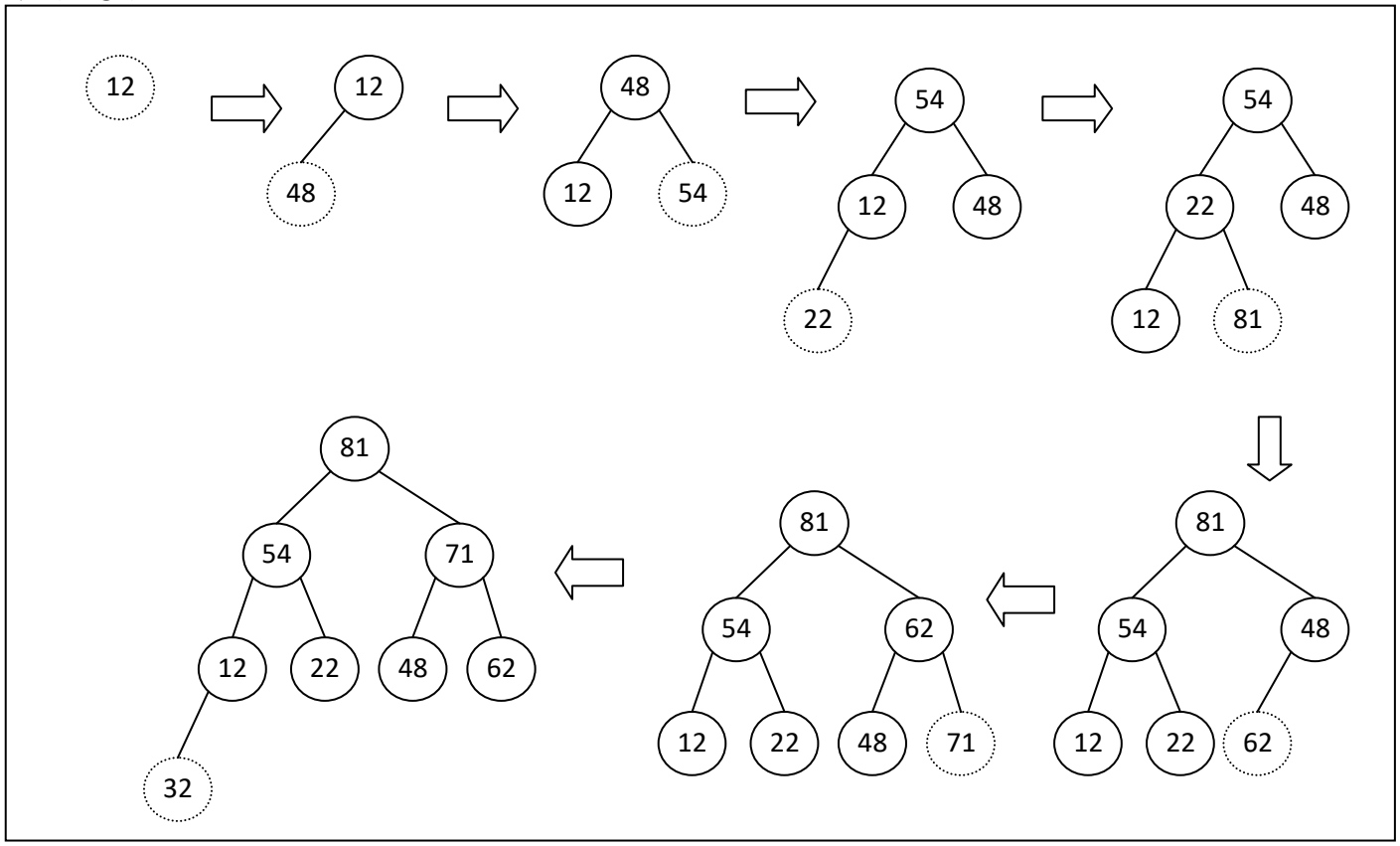
```
int visina(cvor * korijen) {
   int visinaLijevo, visinaDesno;
   if (korijen == NULL)
          return 0;
   visinaLijevo = visina(korijen-> lijevo_dijete);
   visinaDesno = visina(korijen-> desno_dijete);
   if (visinaLijevo >= visinaDesno)
          return 1 + visinaLijevo;
   else
          return 1 + visinaDesno;
}

int balans(cvor *korijen){
   int lijevo, desno;
   if (korijen == NULL)
          return 1;
   lijevo = visina(korijen-> lijevo_dijete);
   desno = visina(korijen-> desno_dijete);
   return abs(lijevo - desno) <= 1 && balans(korijen->lijevo_dijete)
                && balans(korijen-> desno_dijete);
}
```

2. zadatak (10 bodova)

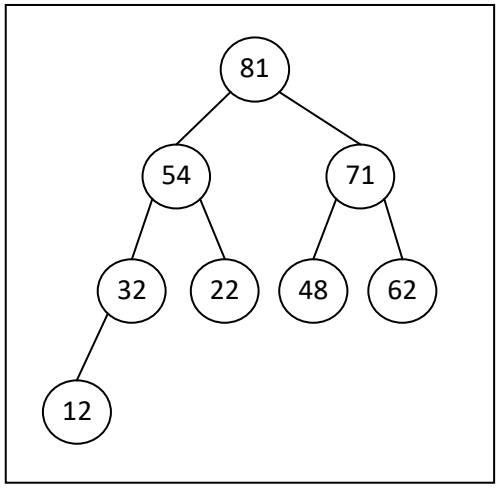
a)*O(n)*



b) *O(nlogn)*



Konačno rješenje:



3. zadatak (10 bodova)

```
#define VELIME 20
#define VELPREZIME 30
#define M 10000

typedef struct {
   int sifra;
   char ime[VELIME + 1];
   char prezime[VELPREZIME + 1];
} zapis;

int trazi(zapis* hash, int * brojPreljeva) {
   int brojac[M] =  {0};
   int brojacPreljeva[M] = {0};
   int zeljenaAdresa;
   int maxVrijednost;
   int maxAdresa;
   int i;
   for (i = 0; i < M; i++) {
         if (hash[i].sifra != 0) {
               zeljenaAdresa = adresa(hash[i].sifra);
               brojac[zeljenaAdresa]++;
               if (i != zeljenaAdresa) brojacPreljeva[zeljenaAdresa]++;
         }
   }
   maxAdresa = 0;
   maxVrijednost = brojac[maxAdresa];
   for (i = 1; i < M; ++i) {
         if (brojac[i] > maxVrijednost) {
               maxVrijednost = brojac[i];
               maxAdresa = i;
         }
   }
   *brojPreljeva = brojacPreljeva[maxAdresa];
   return maxAdresa;
}
```

4. zadatak (10 bodova)

```
Red *IzdvojiVisekratnike(Red *red, int broj){

      Red *pom,pom2;
      int element, temp;

      pom=(Red*)malloc(sizeof(Red));
      init_red(pom);
      init_red(&pom2);

      while(skini(&element, red)){
            if (element % broj==0)
                  dodaj(element, pom);
            else  dodaj(element, &pom2);
      }
      while(skini(&element, &pom2)){
            dodaj(element, red);
      }
      return pom;
}

int main (){
      Red red, *novi_red;

      init_red(&red);
      /*... napuni red i učitaj broj... */
      novi_red = IzdvojiVisekratnike(&red, 3);
      ...
}
```