STABLA I GOMILE

Zavrsni 2014.

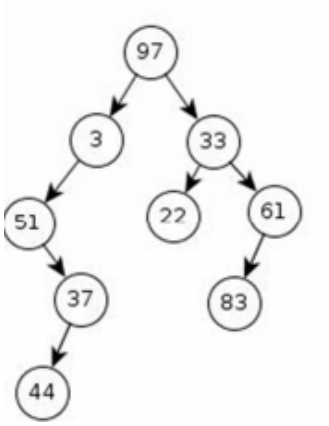
Zadatak 3. (6 bodova)

Binarno stablo jednoznačno je zadano svojim *preorder* i *inorder* ispisima. U nacrtanom okviru skicirajte stablo čiji su *preorder* i *inorder* ispisi (redom s lijeva na desno)

Preorder: `97, 3, 51, 37, 44, 33, 22, 61, 83`

Inorder: `51, 44, 37, 3, 97, 22, 33, 83, 61`

Zadatak 3. (6 bodova)

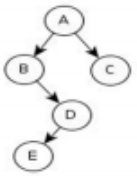


Način formiranja stabla iz danih ispisa:

1. prvi navedeni u preorder-u je glava
2. gledajući u inorder ispisu, očito će sve lijevo od glave biti lijevo podstablo, a sve desno od glave desno podstablo
3. postupak ponavljam rekurzivno za svaki dobiveni podskup

Zadatak 4. (6 bodova)

Čvor stabla definiran je odsječkom:




```
struct cv {
  int vrijednost;
  struct cv *lijevo, *desno;
};
```

Zadani su prototipovi dviju funkcija:

```
int sirinaRazine(cvor* korijen, int razina);
int sirinaStabla(cvor* korijen);
```

Funkcija `sirinaRazine` treba vratiti broj čvorova na razini `razina`. Funkcija `sirinaStabla` treba vratiti najveću širinu svih razina u binarnom stablu, dakle broj čvorova na najširoj razini. Po dogovoru, korijen stabla je na razini 1.

U primjeru na slici širina razine dva je 2, što je najveća širina u ovom stablu pa time i vrijednost koju funkcija `sirinaStabla` mora vratiti.

Napišite funkcije `sirinaRazine` i `sirinaStabla`.

Zadatak 4. (6 bodova)

```
int sirinaRazine(cvor* korijen, int razina) {

	if(korijen==NULL || razina<1)  return 0;
	if(razina==1) return 1;

	return sirinaRazine(
			korijen->lijevo, razina-1) +
		sirinaRazine(
			korijen->desno,razina-1);

}

int sirinaStabla(cvor* korijen) {
	int max=0;
	int razina=1;
	int temp=0;
	do {
		temp=sirinaRazine(korijen, razina);
		++razina;
		max=(max>temp)?max:temp;
	} while (temp>0)
	return max;
}
```

Zadatak 5. (6 bodova)

Zadan je niz brojeva: **36, 58, 44, 28, 96, 62, 76, 38**.

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

b) Počevši od gomile iz podzadatka a), prikažite sortiranje zadanog niza *heapsortom*, jasno prikazujući svaki korak sortiranja (nacrtajte stablo nakon svakog koraka algoritma).

Zadatak 5. (6 bodova)

a)
```
        36
      58 44
   28 96 62 76
 38

        36
      58 44
   38 96 62 76
 28

        36
      58 76
   38 96 62 44
```

```
 28

        36
      96  76
   38 58 62 44
 28

        96
      36  76
   38 58 62 44
 28

        96
      58  76
   38 36 62 44
 28
```

b)

96	58	76	38	36	62	44	28

Zamjena

28	58	76	38	36	62	44	96

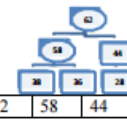
Podešavanje:

76	58	62	38	36	28	44	96

Zamjena

44	58	62	38	36	28	76	96

Podešavanje:



62	58	44	38	36	28	76	96

Zamjena:

28	58	44	38	36	62	76	96

Podešavanje:

58	38	44	28	36	62	76	96

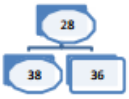
Zamjena

36	38	44	28	58	62	76	96

Podešavanje

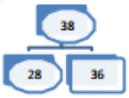
44	38	36	28	58	62	76	96

Zamjena



28	38	36	44	58	62	76	96

Podešavanje



38	28	36	44	58	62	76	96

Zamjena

36	28	38	44	58	62	76	96

Podešavanje – ok
Zamjena

28	36	38	44	58	62	76	96

Zadatak 1. (15 bodova)

Neka su zadani tipovi podataka cvor koji predstavlja čvor binarnog stabla i atom koji predstavlja element reda.

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

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

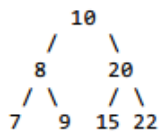
Za rad s redom na raspolaganju su funkcije za inicijalizaciju reda, dodavanje i skidanje elementa iz reda. Funkcije za dodavanje i skidanje elementa iz reda vraćaju 1 ako su se izvršile uspješno.

```
void init_red(Red *red);
int dodajURed(cvor element, Red *red);
int skiniIzReda(cvor *element, Red *red);
```

Napišite **nerekurzivnu** funkciju koja će ispisati postojeće **puno** binarno stablo tako da sve čvorove na istoj razini ispisuje u jednom retku, a svaku razinu u novom retku. Prototip tražene funkcije je:

```
void ispisBezRekurzije(cvor* korijen);
```

Npr. za stablo prikazano na lijevoj strani funkcija mora ispisati čvorove stabla na način prikazan na desnoj strani (prvi redak u primjeru ispisa predstavlja redne brojeve stupaca na zaslonu):




```
01234567890123456789
  10
   8  20
   7   9  15  22
```

Napomene:

U implementaciji tražene funkcije može se koristiti red (ili više njih). Za rad s redom dozvoljeno je koristiti isključivo navedene funkcije. Rekurzivna rješenja neće se priznati.

Zadatak 1. (15 bodova)

```
void ispisBezRekurzije(cvor* korijen) {
Red red;
cvor cv;
int razina = 1, brojCvorova = 0;

if(korijen == NULL) {
  return;
}

init_red(&red);
dodajURed(*korijen, &red);

while(skiniIzReda(&cv, &red)) {
    printf("%4d ", cv.element);
    brojCvorova++;
    if(cv.lijevo) {
      dodajURed(*(cv.lijevo), &red);
    }
    if(cv.desno) {
      dodajURed(*(cv.desno), &red);
    }
    if(brojCvorova >= pow(2, razina - 1)) {
      razina++;
      brojCvorova = 0;
      printf("\n");
    }
  }
}
```

Zadatak 4. (18 bodova)

Čvor binarnog stabla za pretraživanje je struktura:

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

Napisati rekurzivnu funkciju `nti` koja pronalazi n-ti po veličini član stabla. Prototip funkcije je:

```
int nti( int n, cvor *cv, int *rez )
```

pri čemu je `n` redni broj po veličini elementa kojeg tražimo (1 za najveći, 2 za drugi najveći itd.), a `rez` je lokacija na koju treba pohraniti rezultat. Pretpostaviti da je broj `n` uvijek manji od broja čvorova stabla.

Zadatak 4. (18)

```
int nti( int n, cvor *cv, int *rez ) {
        int desno, lijevo;
        if( cv == NULL )
                return 0;
        desno = nti( n, cv->d, rez );

        if( desno == n - 1 ) {
                *rez = cv -> vrijednost;
        }
        lijevo = nti( n - desno - 1
                , cv -> l, rez );
        return desno + lijevo + 1;
}
```

Zadatak 2. (10 bodova)

Napisati rekurzivnu funkciju koja će zrcalno obrnuti binarno stablo, tako da korijen stabla ostane korijen, a lijevo i desno podstablo zamijene mjesta u svakoj razini. Neka je čvor stabla zadan tipom cvor:

```
typedef struct st_cvor {
	KORISNICKI_TIP vrijednost;
	struct st_cvor *l, *d;
} cvor;
```

Zadatak 2. (10)

```
void okreni( cvor *korijen ) {
	cvor *temp;
	if( korijen ) {
		okreni( korijen -> lijevo );
		okreni( korijen -> desno );
		temp = korijen -> lijevo;
		korijen -> lijevo = korijen -> desno;
		korijen -> desno = temp;
	}
}
```

Zadatak 4. (18 bodova)

Binarno stablo sadrži pozitivne cijele brojeve, a čvor je definiran strukturom:

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

Treba napisati rekurzivnu funkciju koja će vratiti koliko unutarnjih čvorova u stablu ima veći prosjek elemenata u lijevom podstablu nego u desnom podstablu i koliko unutarnjih čvorova u stablu ima veći prosjek elemenata u desnom podstablu nego u lijevom.

Zadatak 4. (18)

```
void zaProsjek (cvor *korijen, int *broj, int *zbroj) {
  if (korijen != NULL) {
        zaProsjek(korijen->lijevo,broj,zbroj);
        zaProsjek(korijen->desno,broj,zbroj);
      (*broj)++;
        (*zbroj)+=korijen->vrijednost;
  } else {
    return;
  }
}

void prebrojiStanje(cvor *korijen, int *sL, int *sD){
    int zbroj, broj;
    float prosjekL, prosjekD;

    if (korijen==NULL) return;
    // pogledaj rekurzivno lijevo i desno podstablo
    prebrojiStanje(korijen->lijevo,sL,sD);
    prebrojiStanje(korijen->desno,sL,sD);
    // sada odredi je li veći prosjek lijevo ili desno
    broj=0; zbroj=0;
    zaProsjek(korijen->lijevo, &broj, & zbroj);
    if (broj!=0){
       prosjekL=((double) zbroj)/broj;
    }
    else prosjekL=0; // signal da nema elemenata u podstablu

    broj=0; zbroj=0;
    zaProsjek(korijen->desno, &broj, & zbroj);
    if (broj!=0){
          prosjekD=((double) zbroj)/broj;
    }
    else prosjekD=0;

    if (prosjekL > prosjekD) {
       (*sL)++;
    }
    else if(prosjekD > prosjekL){
       (*sD)++;
    }
}
```

Zavrsni 2013.

Zadatak 3. (6 bodova)

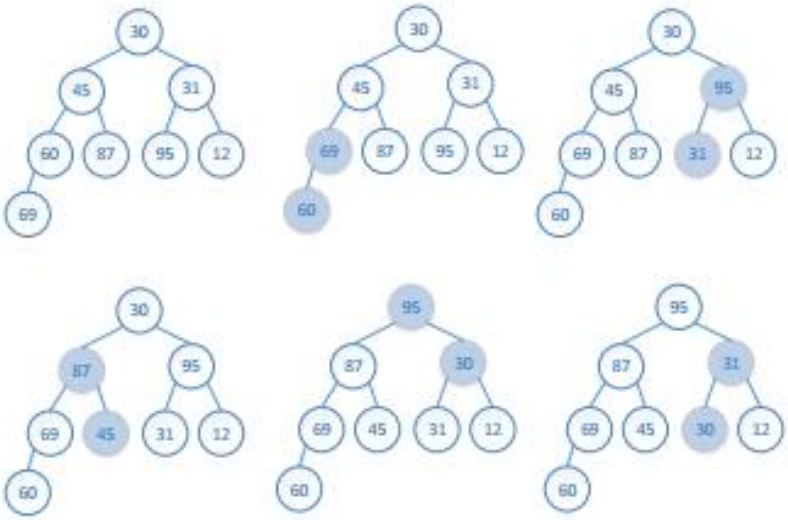
Zadan je niz brojeva: **30, 45, 31, 60, 87, 95, 12, 69**

a) **(3 boda)** Ilustrirajte (nacrtajte stablo nakon svake promjene) stvaranje gomile od zadanog polja brojeva algoritmom čija je složenost za najgori slučaj *O(n)*.

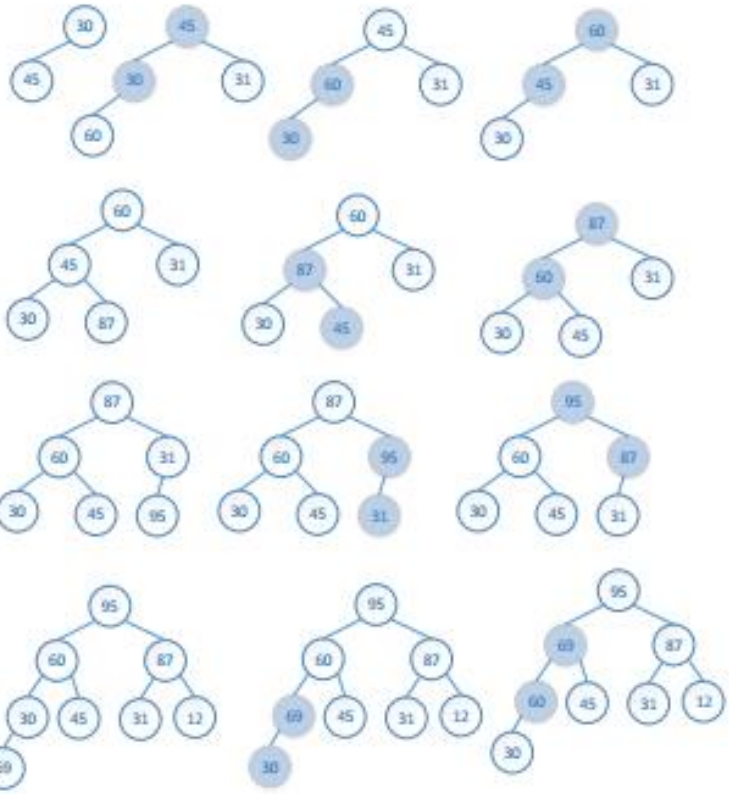
b) **(3 boda)** Ilustrirajte (nacrtajte stablo nakon svake promjene) stvaranje gomile od zadanog polja brojeva algoritmom čija je složenost za najgori slučaj *O(nlogn)*.

Zadatak 3. (6 bodova)

a)



b)



Zadatak 4. (6 bodova)

Binarno stablo za pretraživanje organizirano je po pravilu da su brojevi u lijevom podstablu manji od brojeva u desnom. **Postorder** ispis binarnog stabla za pretraživanje je:

3, 5, 4, 1, 8, 15, 16, 11, 7, 6

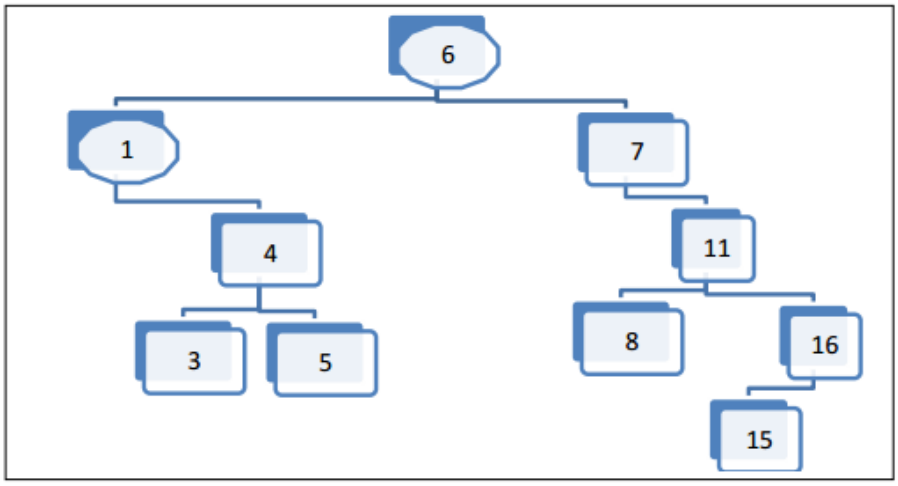
a) **(2 boda) Rekonstruirajte** binarno stablo za pretraživanje (nacrtajte izgled) u prostor s desna

Izgled stabla:

b) **(4 boda)** Binarno stablo za pretraživanje, zadano postorder ispisom, ostvareno je statičkom strukturom polje, pri čemu se prvi element polja (indeks nula) ne koristi, nego je korijen stabla na indeksu 1. Napišite, u odgovarajućem redoslijedu, niz indeksa (i_1, i_2, i_3, ... i_{10}) u polju na kojima se nalaze elementi ispisani postorder obilaskom stabla. Na primjer, i_3 je indeks na kojem se u polju nalazi 3. po redu ispisani element stabla. Odgovor napišite na crtu ispod.

Zadatak 4. (bodova)

a)



b) (10,11,5,2,14,30,15,7,3,1)

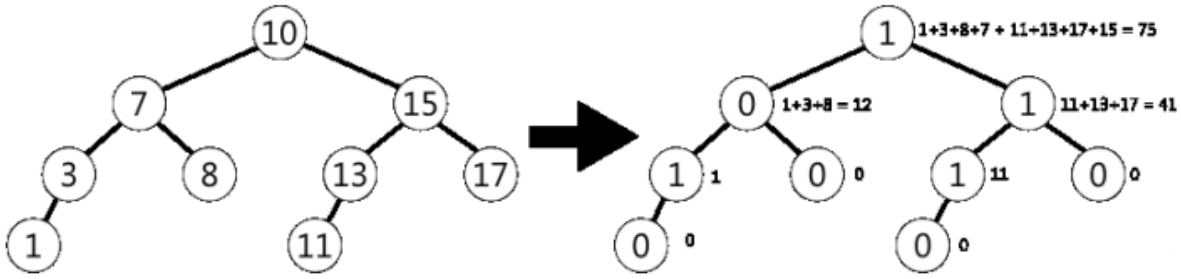
Zadatak 2. (14 bodova)

Binarno stablo sadrži cjelobrojne elemente, a čvor je definiran sljedećom strukturom:

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

Napisati funkciju koja će u svakom čvoru zamijeniti vrijednost sa 0 ako je zbroj svih potomaka tog čvora paran odnosno sa 1 ako je zbroj svih potomaka neparan.

Primjer očekivane transformacije:



Zadatak 2. (14 bodova)

```
int fun(cvor *korijen){
  int suma;
  int trenutnaVrijednost;

  if(korijen == NULL){
    return 0;
  }

  suma = fun(korijen->lijevo) + fun(korijen->desno);
  trenutnaVrijednost = korijen->vrijednost;
  korijen->vrijednost = suma % 2;
  return suma + trenutnaVrijednost;
}
```

Zadatak 3. (14 bodova)

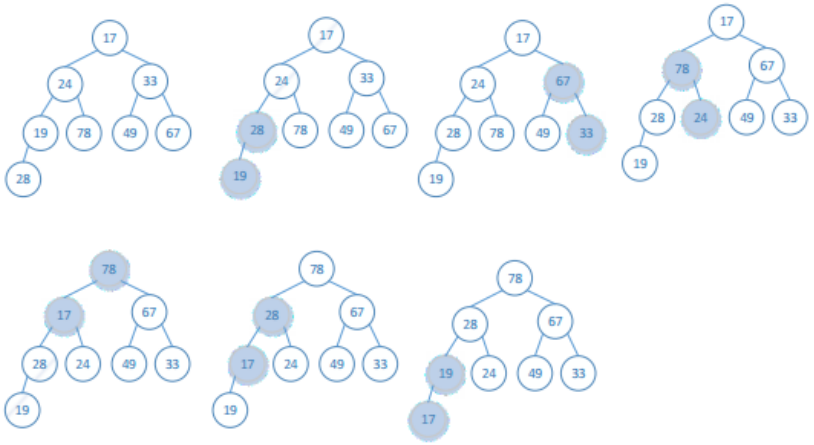
Zadan je niz brojeva: **17, 24, 33, 19, 78, 49, 67, 28**.

a) **(7 bodova)** Ilustrirajte (nacrtajte stablo nakon svake promjene) stvaranje gomile s relacijom ***veći od*** (max heap) od zadanog polja brojeva algoritmom čija je složenost za najgori slučaj *O(n)*.

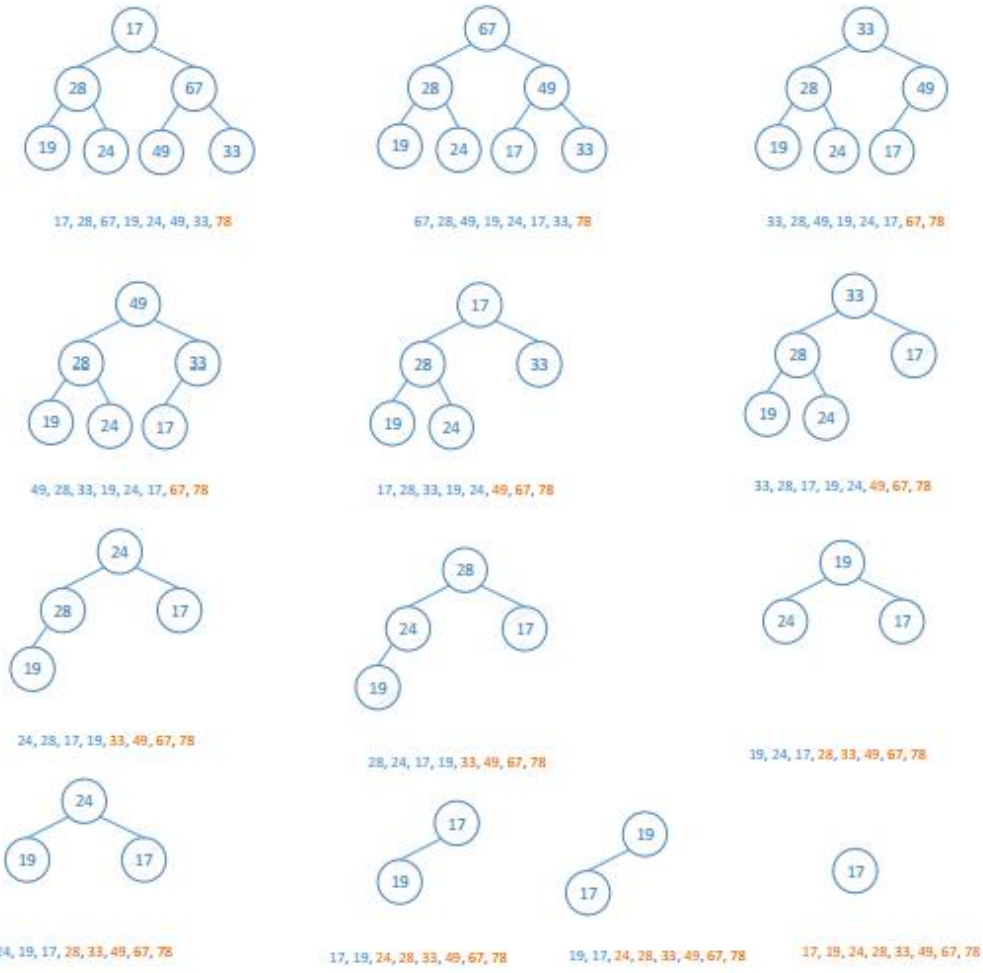
b) **(7 bodova)** Za gomilu iz a) zadatka prikažite postupak uzlaznog *heapsorta*. Prikažite svaki korak sortiranja (nacrtajte stablo nakon svake zamjene dva elementa).

Zadatak 3. (14 bodova)

a)



b)



Zadatak 3. (19 bodova)

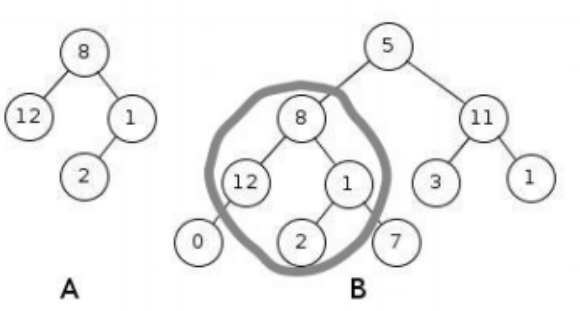
Binarno stablo sadrži cjele brojeve, a čvor je definiran sljedećom strukturom:

```
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.

Primjer: 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.

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;
}
```

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:

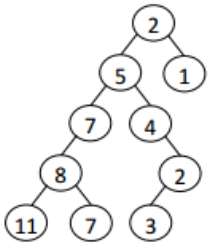
```
typedef struct cv {
  int el;
  struct cv *lijevo;
  struct cv *desno;
} cvor;
```

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:

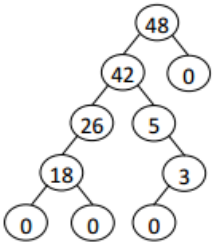
```
cvor *dodaj (cvor *korijen, int broj);
```

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



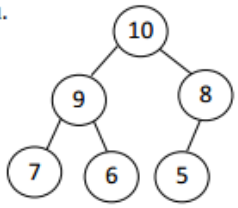
Slika 2

Zadatak 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. (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.

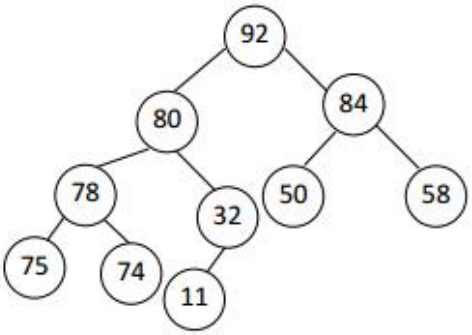


b)*(2 boda)* Zadana je gomila kao na slici.
Prikažite sve korake uzlaznog heap sorta

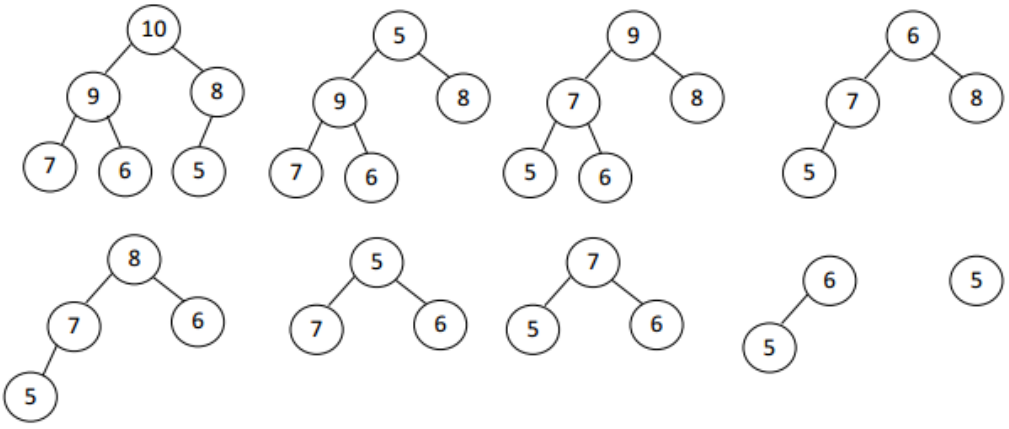
Zadatak 3. (2 + 2 boda)

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

a)



b)



3. zadatak (15 bodova)

Binarno stablo sadrži podatke o artiklima koji su predstavljeni šifrom (pozitivan cijeli broj) i cijenom (struktura **el**), dok su čvorovi binarnog stabla definirani strukturom **cvor** :

```
typedef struct{
      int sifra;
      float cijena;
} el;
```

```
typedef struct s {
      el element;
      struct s *lijevo, *desno;
} cvor;
```

Uz pretpostavku da je stablo sortirano po šifri artikla: lijevo artikli s manjom šifrom, desno oni s većom te da u stablu ne postoje dva artikla s istom šifrom:

a) Napišite rekurzivnu funkciju **razina** koja će za zadanu šifru ispisati na kojoj razini unutar stabla se nalazi artikl s tom šifrom. Funkcija vraća 0 ako ne postoji artikl sa zadanom šifrom. Prototip funkcije je:
```
int razina(int sifra, cvor *korijen)
```

b) Napišite **rekurzivnu** funkciju **zbrojCijenaNaRazini** koja vraća zbroj cijena svih artikala koji se nalaze na zadanoj razini. Prototip funkcije je:
```
float zbrojCijenaNaRazini (int razina, cvor *korijen)
```

3. **(15 bodova)**

a) (9 bodova)

```
int razina(int sifra, cvor *korijen)
{
      int d;
      if (!(korijen)) return 0;
      if (sifra==korijen->element.sifra){
            return 1;
      }
      else{
            if (sifra<korijen->element.sifra){
                  d=razina(sifra, korijen->lijevo);
            }
            else d=razina(sifra,korijen->desno);
            if (d) return 1+d;
            return 0;
      }
}
```

b) (6 bodova)

```
float zbrojCijenaNaRazini (int razina, cvor *korijen){
      if (!korijen) return 0;
      if (razina<1) return 0;
      if (razina==1) return korijen->element.cijena;
      return zbrojRazina(razina -1,korijen->lijevo)
           +zbrojRazina(razina -1,korijen->desno);
}
```

3. zadatak (10 bodova)

Napisati funkciju koja će na zaslon ispisati vrijednost iz svih čvorova binarnog stabla koji imaju neparni broj potomaka. Dozvoljen je samo jedan prolaz kroz stablo. Čvor stabla zadan je strukturom:

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

Prototip funkcije je: **`int ispisi (cvor *korijen);`**

3. (14 bodova)

```
int ispisi(cvor *korijen){
        int brPotomaka = 0;
        if(korijen == NULL){
                return 0;
        }
        brPotomaka = ispisi(korijen->lijevo) + ispisi(korijen->desno);
        if(brPotomaka % 2){
                printf("Cvor %d ima neparan broj potomaka - %d\n", korijen->vrijednost, brPotomaka);
        }
        return brPotomaka + 1;
}
```

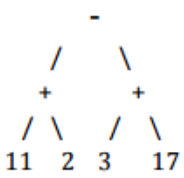
5. zadatak (12 bodova)

U binarnom stablu pohranjena je aritmetička operacija tako da svaki čvor sadrži ili operand ili operator. I operand i operator pohranjeni su kao niz znakova. Operandi su uvijek cijeli brojevi i nalaze se u listovima stabla, a od operatora postoji samo zbrajanje (+) i oduzimanje (-).
Za pretvaranje niza znakova u cijeli broj (int) možete napraviti posebnu funkciju.
Čvor stabla zadan je strukturom:

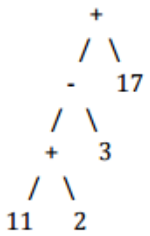
```
typedef struct cvor{
      char *elem;
      struct cvor *lijevo;
      struct cvor *desno;
} cvor;
```

Napisati funkciju koja u pozivajući program vraća rezultat te aritmetičke operacije.

Primjer 1: (11+2)-(3+17) = -7



Primjer 2: (((11+2) -3)+17) = 27



Zad 5. (12 bodova)

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>

typedef struct cvor{
	char *elem;
	struct cvor *lijevo;
	struct cvor *desno;
} cvor;

//Niz znakova koji sadrži broj pretvara u int
int str_to_int(char *str){
	int len = strlen(str);
	int i=0, num=0;
	for(i=len-1; i>=0; i--){
		num += (str[i] - 48) * pow(10., len-i-1);
	}
	return num;
}

int racunaj(cvor *korijen){
	int rez = 0;
	if(korijen->lijevo == NULL && korijen->desno == NULL){
		//List stabla - pretvaramo u int
		return str_to_int(korijen->elem);
	}else{
		if(korijen->elem[0] == '+'){
			//Operacija zbrajanja
			return racunaj(korijen->lijevo) + racunaj(korijen->desno);
		}else if(korijen->elem[0] == '-'){
			//Operacija oduzimanja
			return racunaj(korijen->lijevo) - racunaj(korijen->desno);
		}
	}
}
```