

Zadaci za samostalnu vježbu za 3. blic

ZADACI:

Jednostruko i dvostruko povezane liste

Zadatak 1

Napišite funkciju za dodavanje elemenata na stog realiziran jednostruko povezanom listom ako je zadan sljedeći prototip funkcije:

```
int dodaj (cvor **vrh, int element);
```

Zadatak 2

U jednostruko povezanu listu spremaju se zapisi sljedećeg tipa:

```
typedef struct s{
    int pbr;                // poštanski broj
    char mjesto[100+1];    // naziv mjesta
    struct s *sljed;
} zapis;
```

Kako glasi funkcija koja pronalazi u listi zapis o mjestu sa zadanim poštanskim brojem i vraća taj zapis u glavni program. Ako takav zapis ne postoji u listi funkcija vraća NULL. Lista nije sortirana.

Stabla

Zadatak 1

Napisati **nerekurzivnu** funkciju koja vraća razinu na kojoj se nalazi najveći element u stablu. U stablu su pohranjeni cjelobrojni elementi i stablo je sortirano (lijevo manji, desno veći).

Zadatak 2

Napisati **rekurzivnu** funkciju koja vraća razinu na kojoj se nalazi najveći element u stablu. U stablu su pohranjeni cjelobrojni elementi i stablo je sortirano (lijevo manji, desno veći).

Gomila kao prioritetni red, heapsort

Zadatak 1

Kako izgleda ispis gomile po razinama ako je gomila formirana za ulazni niz 25 35 12 40 1 90 7 15 algoritmom čija je složenost za najgori slučaj $O(n)$?

Zadatak 2

Zadana je gomila koja je pohranjena u polju:

88	66	77	33	55	44	11	22
----	----	----	----	----	----	----	----

Prikažite postupak uzlaznog heapsorta.

RJEŠENJA:

Jednostruko i dvostruko povezane liste

Zadatak 1

```
int dodaj (atom **vrh, int element) {
    cvor *novi;
    if ((novi= (atom *) malloc(sizeof(cvor))) == NULL) return 0;
    novi->element = element;
    novi->sljed = *vrh;
    *vrh = novi;

    return 1;
}
```

Zadatak 2

```
zapis *nadji(zapis *glava, int pbr) {
    while (glava && (glava->pbr != pbr))
        glava = glava->sljed;
    return glava;
}
```

Stabla

Zadatak 1

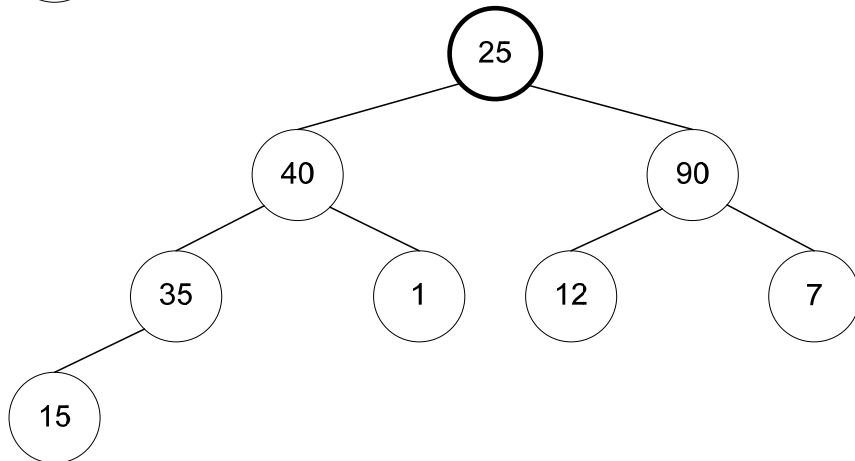
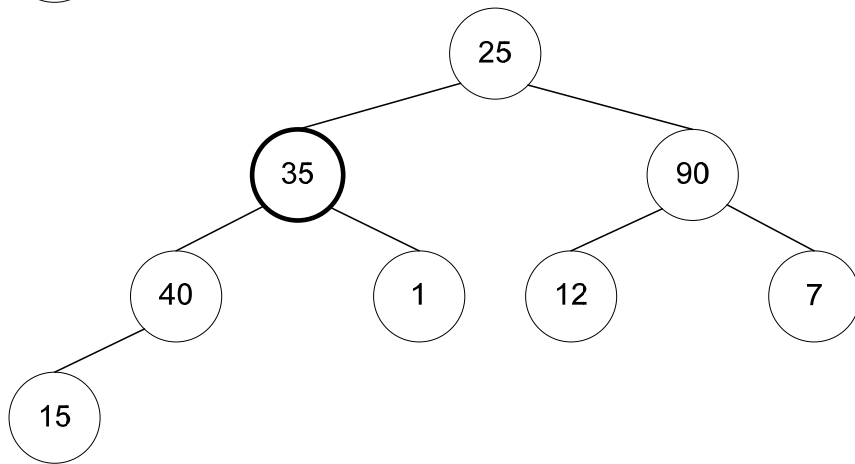
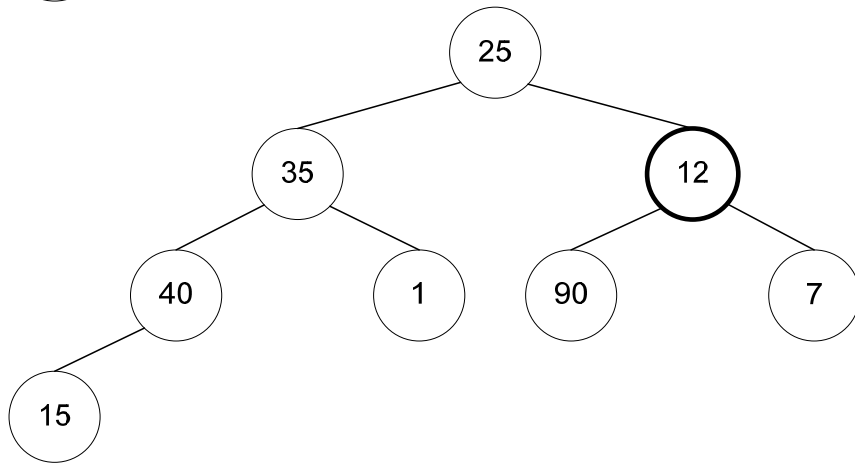
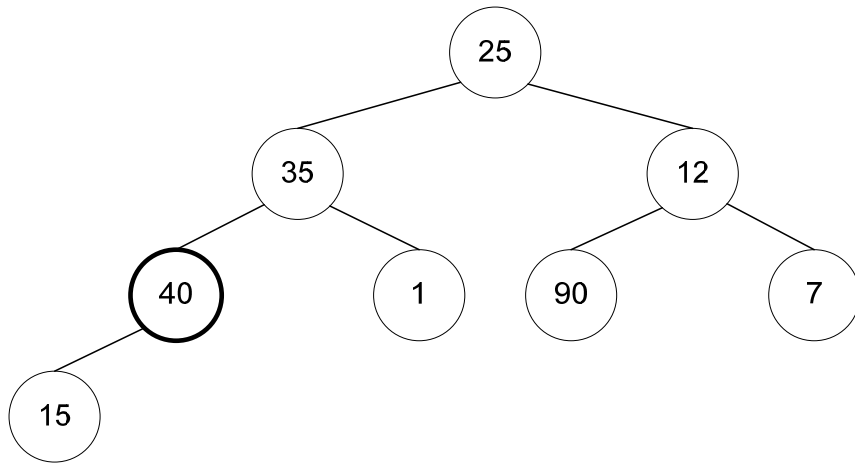
```
int rNajveci(cvor *korijen)
{
    int razina=0;
    while(korijen)
    {
        korijen=korijen->d;
        razina++;
    }
    return razina;
}
```

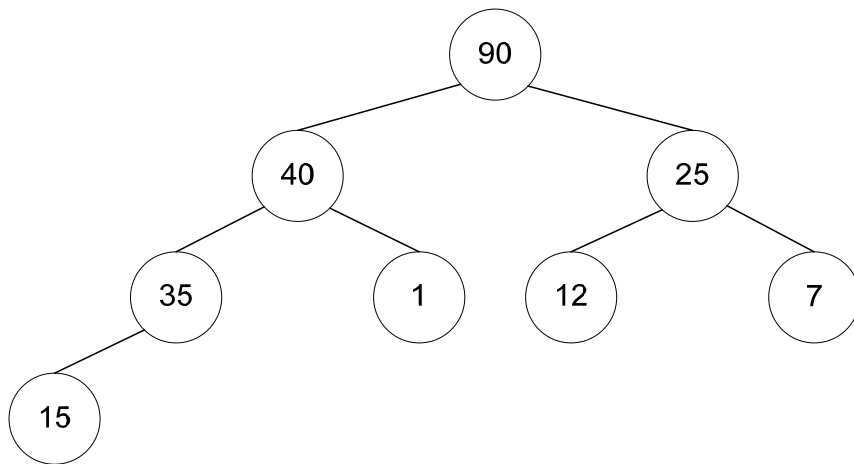
Zadatak 2

```
int rNajveciRek(cvor *korijen)
{
    if(!korijen)
        return 0;
    else
        return (1+rNajveciRek(korijen->d));
}
```

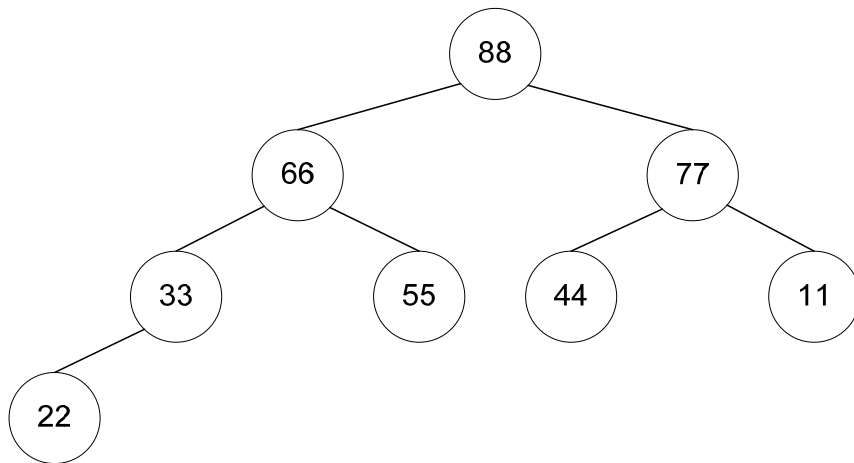
Gomila kao prioritetni red, heapsort

Zadatak 1

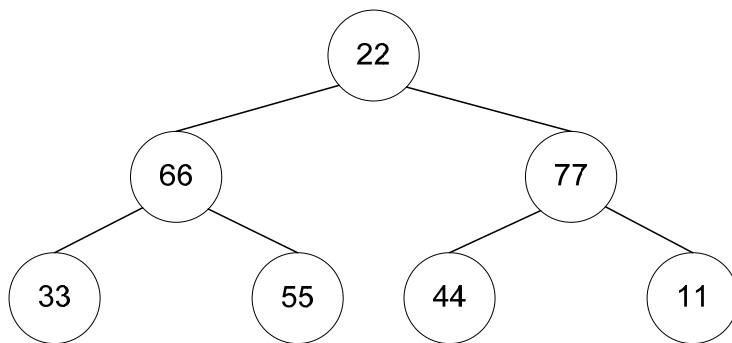




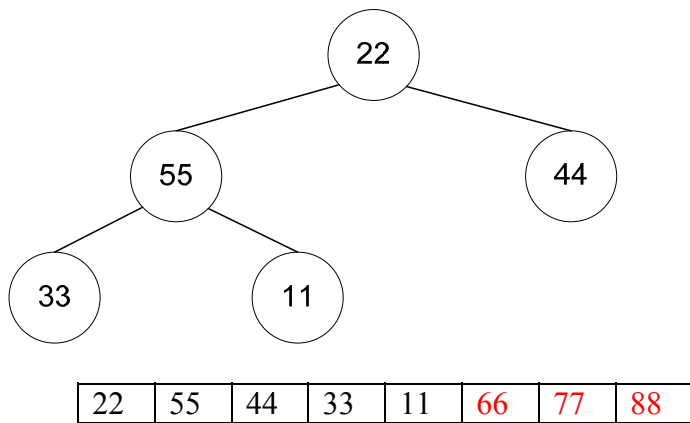
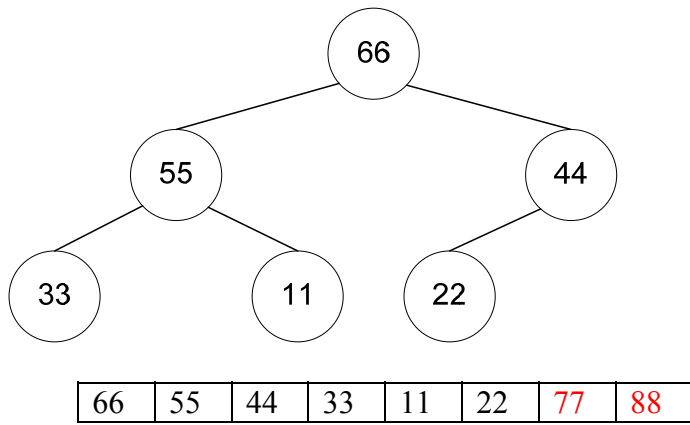
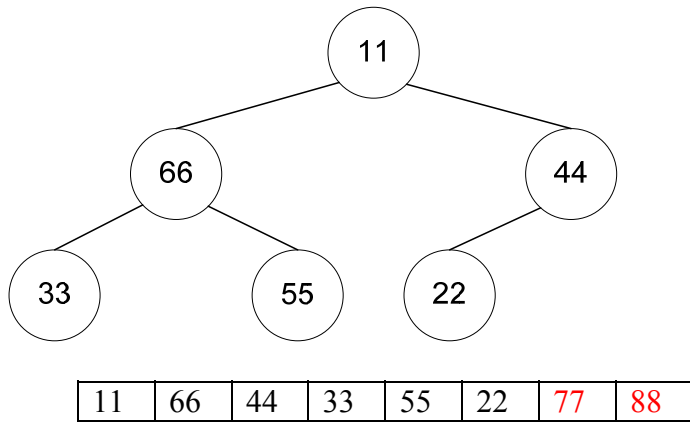
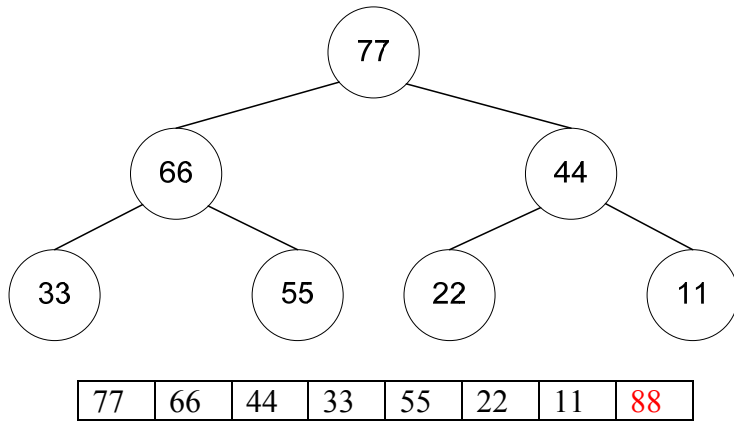
Zadatak 2

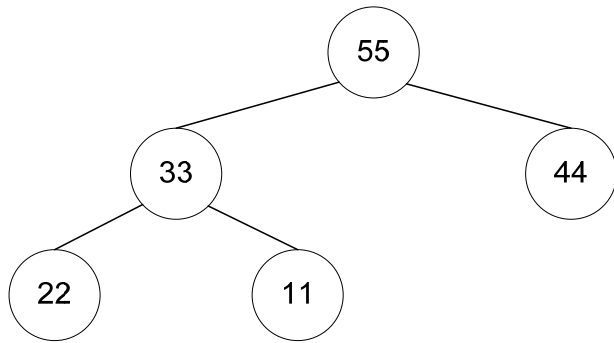


88	66	77	33	55	44	11	22
----	----	----	----	----	----	----	----

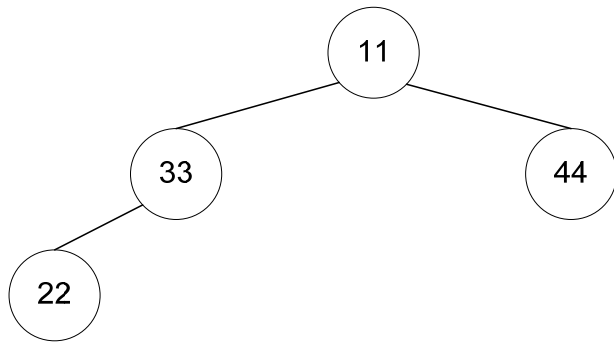


22	66	77	33	55	44	11	88
----	----	----	----	----	----	----	----

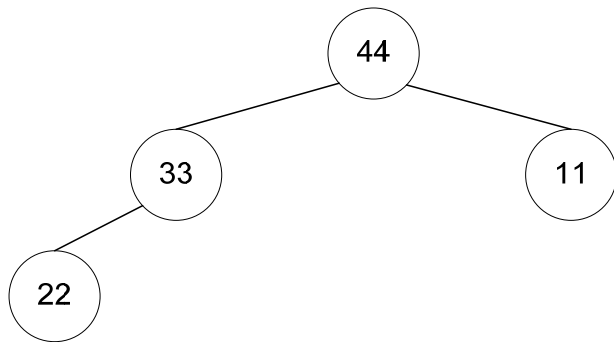




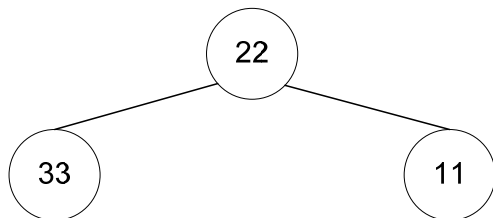
55	33	44	22	11	66	77	88
----	----	----	----	----	----	----	----



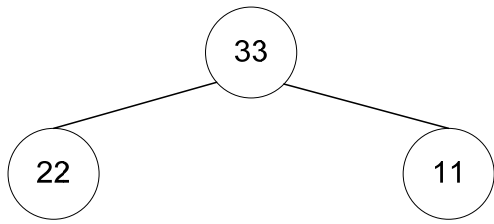
11	33	44	22	55	66	77	88
----	----	----	----	----	----	----	----



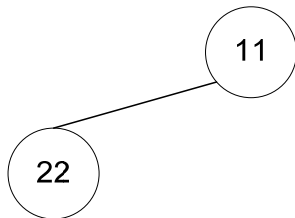
44	33	11	22	55	66	77	88
----	----	----	----	----	----	----	----



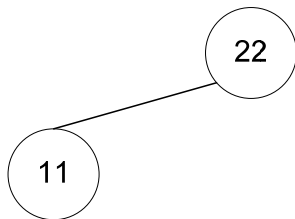
22	33	11	44	55	66	77	88
----	----	----	----	----	----	----	----



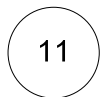
33	22	11	44	55	66	77	88
----	----	----	----	----	----	----	----



11	22	33	44	55	66	77	88
----	----	----	----	----	----	----	----



22	11	33	44	55	66	77	88
----	----	----	----	----	----	----	----



11	22	33	44	55	66	77	88
----	----	----	----	----	----	----	----

11	22	33	44	55	66	77	88
----	----	----	----	----	----	----	----