Tietorakenteet, laskuharjoitus 5, 14.-18.2.

1. Alunperin tyhjään binäärihakupuuhun lisätään monisteen insert-operaatiota käyttäen avaimia seuraavassa järjestyksessä: 2, 0, 3, 7, 9, 1, 5, 6 ja 8. Näytä miten puu kasvaa lisäysten seurauksena. Piirrä siis kuva tilanteesta jokaisen (tai ainakin melkein jokaisen) lisäyksen jälkeen.

   Simuloi delete-operaation toimintaa tuloksena olevevalle puulle parametrina solmu, jossa avain 3. Poista tämän jälkeen avain 9. Tämän jälkeen avain 2. Ja lopuksi avain 5. Piirrä jokaisen operaation jälkeen tuloksena oleva puu.

2. Toteuta Javalla binäärihakupuu, jossa on operaatiot insert ja search. Puuhun talletettavien avainten tyyppinä on `long` (tai `int`, ensi viikolla voi olla hyötyä longista mutta muutos on helppo). **Tätä tehtävää varten puuhun ei kannata laittaa parent-viitteitä ollenkaan!**

   Testaa puusi toimintaa. Testaamista varten on hyödyllistä toteuttaa algoritmi, joka tulostaa koko puun sisällön. Yksinkertaisimmillaan puun sisällön voi tulostaa esim. käymällä solmut läpi sisäjärjestyksessä (ks. monisteen sivut 151-156), tällöin solmujen pitäisi tulostua suuruusjärjestyksessä.

   "Kehittyneempi" visualisointi puulle tehdään tehtävässä 5.

   Yksi tapa toteuttaa rekursiiviset metodit olio-orientoituneessa koodissa on tehdä tietorakennetta edustavalle luokalle "normaali"ulospäin näkyvä metodi joka kutsuu puun sisäistä rekursiivista metodia, esim. seuraavaan tyyliin:

```
public void tulostaJarjestyksessa(){
    tulostaJarjestyksessa( root );
}

private void tulostaJarjestyksessa(Puusolmu solmu) {
    if ( solmu == null ) return;

    tulostaJarjestyksessa(solmu.left);
    System.out.println( solmu.key );
    tulostaJarjestyksessa(solmu.right);
}
```

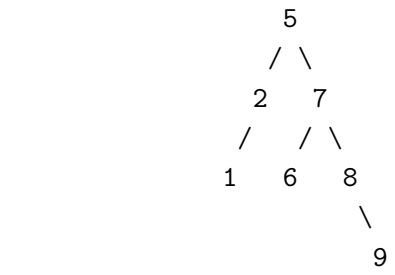
   Huom: seuraavat tehtävät voit tehdä joko pseudokoodina tai ohjelmoida edellisen tehtävän puuhusi. Tehtävistä 3 ja 5 perustuvat monisteen sivuilla 151-156 esitetyihin puun läpikäynteihin.

3. Tee algoritmi, joka laskee puun lehtisolmujen lukumäärän. Mikä on algoritmisi aika- ja tilavaativuus?

4. Esitä algoritmi, joka saa syötteekseen binääripuun ja tulostaa solmut tasojärjestyksessä, eli ensin tason 0 solmut, sitten tason 1 solmut, tason 2. solmut ...

Mikä on algoritmisi aika- ja tilavaativuus?

Lisäviritys edelliseen (ei vaadita laskarirastiin): Laajenna algoritmiasi siten, että se tekee puulle yksinkertaisen visualisoinnin. Periaate on seuraava, jos puu on esim.



Tulostaa algoritmi

```
_5_
_2  _7_
1  6  8_
9
```

Solmut tulostataan tasoittain. Solmun yhteyteen tulostetaan tieto siitä onko solmulla lapsia. Merkki _ solmun vasemmalla puolella tarkoittaa että solmulla on vasen lapsi ja merkki _ solmun oikealla puolella tarkoittaa, että solmulla on oikea lapsi. Esim. puussa 2:lla on vain vasen lapsi, joten tulostukseen tulee _2.

5. Esitä algoritmi, joka tarkistaa, ovatko kaksi binääripuuta samanlaiset. Algoritmille annetaan viite kummankin binääripuun juurisolmuun.

   Esimerkiksi seuraavassa kuvassa binääripuut A ja B ovat samanlaiset. Binääripuut A ja C eivät ole samanlaiset, koska niiden rakenne on erilainen. Binääripuut A ja D eivät ole samanlaiset, koska solmuissa olevat luvut eivät ole samat.