# Lösningsförslag till tentamen*

| | |
|---|---|
| **Kursnamn** | **Algoritmer och datastrukturer** |
| **Tentamensdatum** | **2009-06-01** |
| **Program** | **DAI2+I2** |
| **Läsår** | **2008/2009, lp IV** |
| **Examinator** | **Uno Holmer** |

* se även `www.chl.chalmers.se/~holmer/` där finns det mesta av kursmaterialet.

**Uppgift 1** (10 p)

Ingen lösning ges. Se kurslitteraturen.

**Uppgift 2** (12 p)

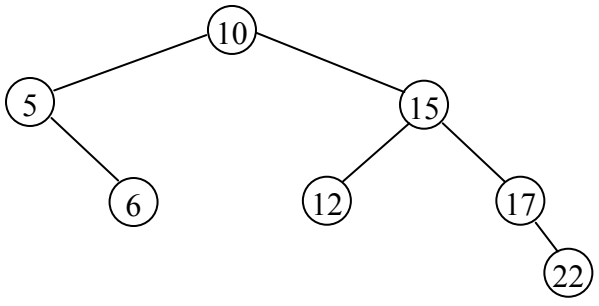
```
public class Section {
    ...

    public void print() {
        print("1");
    }

    private void print(String sectionNumber) {
        System.out.println(sectionNumber + " " + title);
        int i = 1;
        for ( Section s : subsections )
            s.print(sectionNumber + "." + i++);
    }
}
```

**Uppgift 3** (4 p)

På grund av en beklaglig redigeringsmiss föll ett tal bort i den första och den andra sekvensen. Så här skulle de se ut:

10, 20, 6, 15, 4, 5, 30
5, 3, 10, 8, 12, 7
10, 15, 12, 17, 22, 5, 6
10, 5, 7, 3, 15, 20, 1, 18

Sekvensen 10, 15, 12, 17, 22, 5, 6 ger AVL-trädet



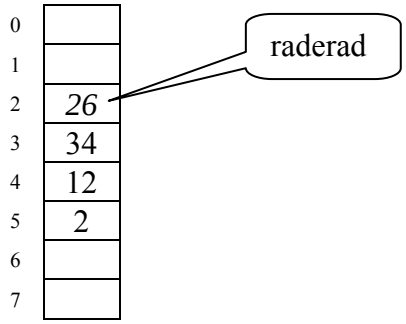

(De felaktiga sekvenserna gav också AVL-träd, vilket de som valt dessa fick poäng för.)

**Uppgift 4** (1+1+4 p)

a) Den första har ej primtalsstorlek.

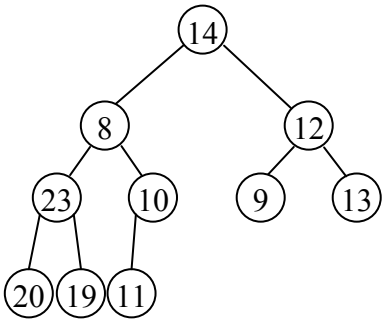
b) I den andra har λ överskridit 0.5.

c)

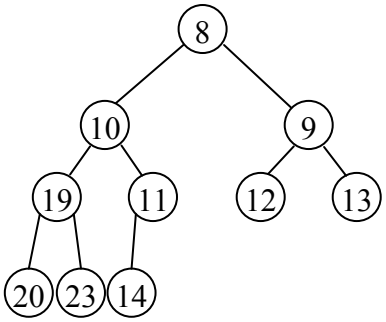



**Uppgift 5** (1+2+4 p)

a)



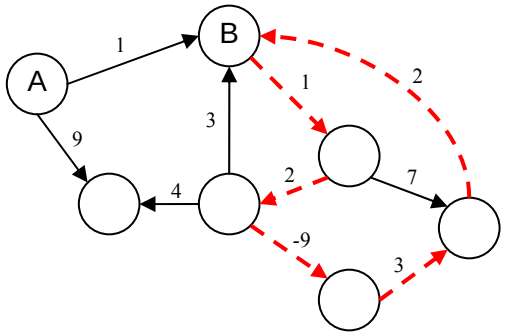

b)




c) insert: AC: O(1), WC: O(log N), buildHeap: AC: O(N), WC: O(N).

## Uppgift 6 (2+6 p)

a)
De markerade bågarna bildar en negativ kostnadscykel. Om Dijkstras metod skulle appliceras på grafen så skulle den försöka minimera avstånden i oändlighet. Om t.ex. A väljs som startnod blir först det preliminärt minimala avståndet till B 1, men efter ett varv i cykeln kan det sänkas till 0, efter ett varv till, -1, o.s.v.




b) ABCDE, ABDCE, ABDEC, BACDE, BADCE, BADEC, BDAEC, BDACE, BDEAC.

## Uppgift 7 (4+9 p)

a) Konstruktorn i `Mobile`:

```
public Mobile( Mobile left, Mobile right ) {
    type = MobileType.COMPOSITE;
    this.left = left;
    this.right = right;
    leftLength =
        ROD_LENGTH*right.weight/(left.weight+right.weight);
    rightLength = ROD_LENGTH - leftLength;
    weight = left.weight + right.weight + ROD_WEIGHT;
}
```

b) Byggmetoden:

```
public static Mobile build(List<Integer> weights) {
    if ( weights.isEmpty() )
        return null;
    else {
        PriorityQueue<Mobile> pq = new PriorityQueue<Mobile>();
        for ( Integer w : weights )
            pq.add(new Mobile(w));

        while ( pq.size() > 1 )
            pq.add(new Mobile(pq.poll(),pq.poll()));

        return pq.poll();
    }
}
```