Lösningsförslag till tentamen

Kursnamn	**Algoritmer och datastrukturer**
Tentamensdatum	**2013-08-27**
Program	**DAI2+I2**
Läsår	**2012/2013, lp 4**
Examinator	**Uno Holmer**

Uppgift 1 (10 p)

Ingen lösning ges. Se kurslitteraturen.

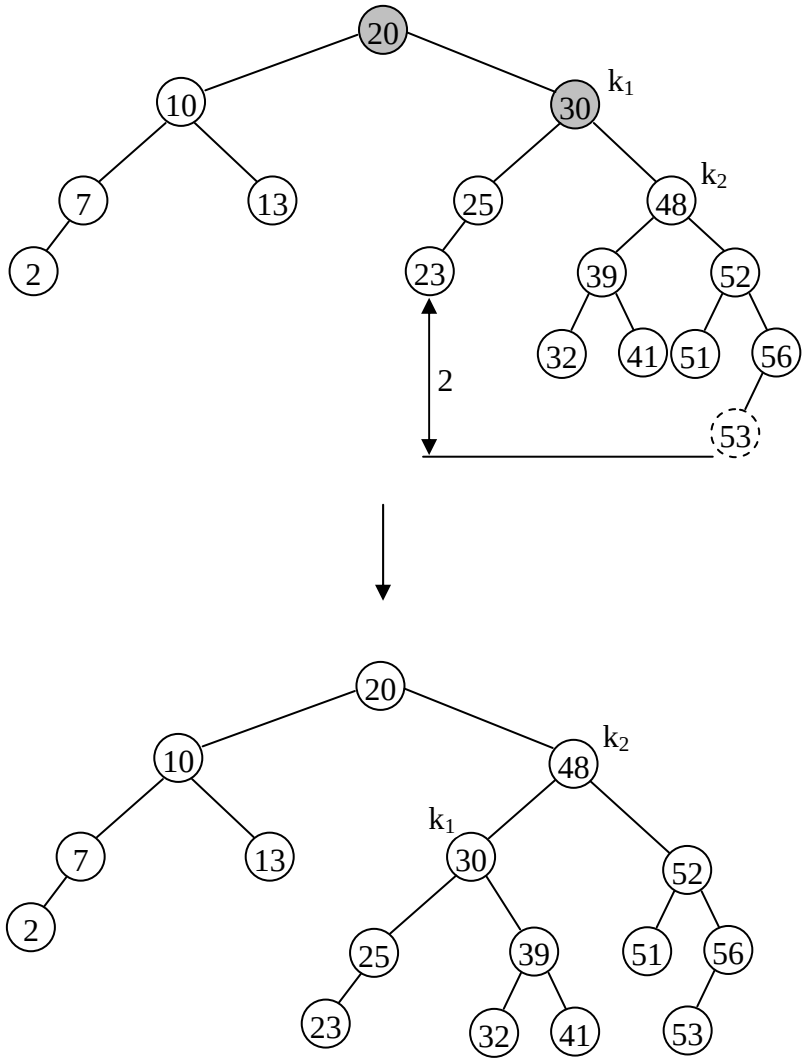
Uppgift 2 (10 p)

```
public static void prettyPrint( TreeNode e ) {
     if ( e != null ) {
          if ( e.nodeType == TreeNode.NodeType.VARIABLE )
               System.out.print(e.value);
          else if ( e.nodeType == TreeNode.NodeType.OPERATOR ) {
               System.out.print("(");
               if ( e.left != null )
                    prettyPrint(e.left);
               System.out.print(e.value);
               prettyPrint(e.right);
               System.out.print(")");
          }
          else
               System.out.print("Unknown node type: " + e.nodeType );
     }
}
```

Uppgift 3 (6+4 p)

a)
Preorder: ABDEGCFHI
Postorder: DGEBHIFCA
Inorder: DBGEAHFIC

b)
När 53 sätts in bryts AVL-villkoret. Den djupaste obalanserad noden är k_1. En enkelrotation enligt nedan upprättar balansen i trädet. Detta motsvarar fall 4 i Weiss:s figur 19.26.


20
10
30
k1
7
13
25
48
k2
2
23
39
52
32
41
51
56
2
53
20
10
48
k2
7
13
k1
30
52
2
25
39
51
56
23
32
41
53

Uppgift 4 (4+3 p)

a)
Så här ser tabellen ut efter anropssekvensen i uppgiften

0	10
1	
2	
3	43
4	
5	
6	
7	
8	
9	
10	21

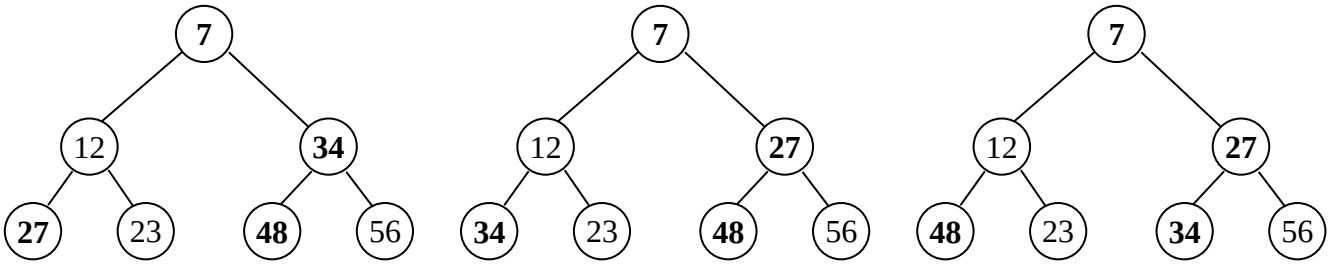
hash(21) = 10
hash(10) = 10
hash(43) = 10

`add(21);`	Elementet sätts in på hashpositionen.
`add(10);`	10 kolliderar med 21 och hamnar i alternativ position $(10 + 1^2) \bmod 11 = 0$.
`add( 43);`	43 kolliderar först med 21 och sen med 10 och hamnar i position $(10 + 2^2) \bmod 11 = 3$.
`remove(10);`	Cell nr 0 markeras som struken
`add(43);`	`43` sätts ej in igen eftersom duplikat ej tillåts i en hashtabell. Observera att 43 ej sätts in i position 0 där 10 ströks tidigare. Alla insättningar måste i princip föregås av en misslyckad sökning. Därför är strukna celler "frysta" tills omhashning sker.

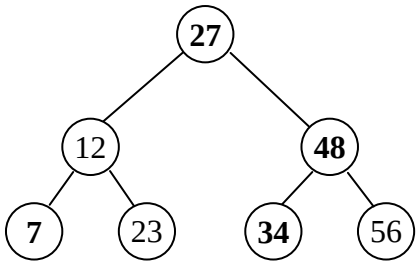
b)
Belastningsfaktorn får ej överskida 0.5 vid kvadratisk sondering. Med 11 platser i tabellen görs omhashning då det sjätte elementet (inklusive strukna) sätts in. Tabellen måste alltid ha primtalsstorlek och det minsta primtalet större än 2*11 är 23.

Uppgift 5 (3+3 p)

a) Binär hög. Vilket som helst av



b) Endast ett binärt sökträd är möjligt.



Uppgift 6 (3+1+1+1+1 p)

a)
g1 kan uteslutas eftersom det finns en båge från A till B och då kan inte en topologisk ordning inledas med BA. Av analogt skäl kan vi utesluta g3 som har en båge från B till A och då kan inte en topologisk ordning inledas med AB. I g4 finns en båge från D till C, vilket utesluter ABCD. Återstår g2 för vilken ABCD och BADC är möjliga ordningar, tillkommer ADBC, ABDC och BACD.

b) Först A, sen BCD i valfri ordning, sist E.

c) ADCBE

d) A(0), B(14), C(12), D(7), E(22).

e) I Dijkstras algoritm används en prioritetskö.

Uppgift 7 (10 p)

Följande algoritm utnyttjar en prioritetskö. Alla tal som läggs i kön är hammingtal men kön kommer att innehålla duplikat, t.ex. 2x3 och 3x2 m.fl.

```
public static void hamming( long n ) {
    PriorityQueue<Long> pq = new PriorityQueue<Long>();
    pq.add( (long)1 );
    long last = -1;
    int i = 0;
    while ( i < n ) {
        long h = pq.poll();
        if ( h != last ) {
            System.out.println( h );
            pq.add( 2*h );
            pq.add( 3*h );
            pq.add( 5*h );
            i++;
        }
        last = h;
    }
}
```