Lösningsförslag till tentamen *P r e l i m i n ä r*

Kursnamn	**Algoritmer och datastrukturer**
Tentamensdatum	**2014-06-03**
Program	**DAI2+I2**
Läsår	**2013/2014, lp 4**
Examinator	**Uno Holmer**

Uppgift 1 (10 p)

Ingen lösning ges. Se kurslitteraturen.

Uppgift 2 (10 p)

```
public static List<String> findPaths(TreeNode t,char x) {
    if ( t == null )
        return new ArrayList<String>();
    else {
        List<String> l1 = findPaths(t.left,x),
                     l2 = findPaths(t.right,x);
        if ( ! l1.isEmpty() )
            prefixAll("0",l1);
        if ( ! l2.isEmpty() )
            prefixAll("1",l2);
        l1.addAll(l2);
        if ( t.element == x )
            l1.add("");
        return l1;
    }
}

private static void prefixAll(String p,List<String> l) {
    for ( int i = 0; i < l.size(); i++ )
        l.set(i,p + l.get(i));
}
```

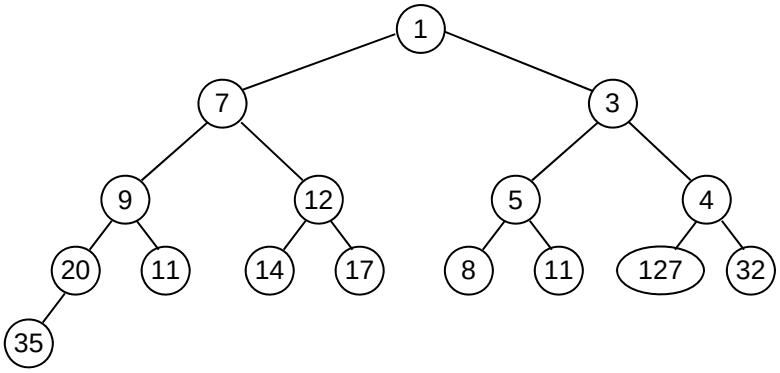
Uppgift 3 (3+4 p)

a) Inorder: DBGEAFC, Preorder: ABDEGCF, Postorder: DGEBFCA

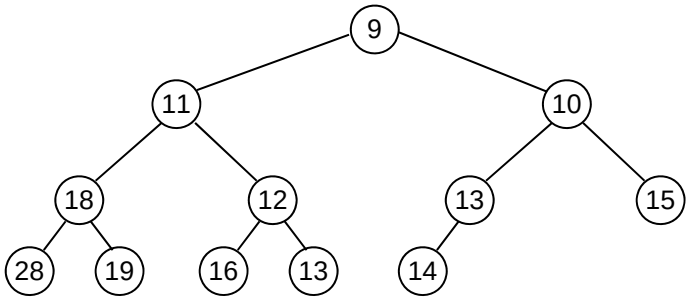
b) Insättning av ett nytt element under 2, 6, 10 eller 14 ger upphov till höjdskillnaden 2 mellan vänster och höger delträd. Vid insättning av t.ex. 1, 3, 5 eller 7 är en enkelrotation tillräcklig (fall 1 i Weiss). Vid insättning av t.ex. 9, 11, 13 eller 15 krävs en dubbelrotation (fall 2 i Weiss).

Uppgift 4 (2+3 p)

a)



b)



Uppgift 5 (5+5 p)

a) I G_1 finns en båge med en negativ kostnad vilket gör att Dijkstras algoritm eventuellt inte terminerar. Den terminerar i det här fallet, men skulle inte göra det om den negativa bågen har kostnaden -24 eller lägre. Kursbokens implementering skulle avbryta beräkningen för G_1. För G_2 är de kortaste viktade avstånden från nod F: F-A:5, F-B:13, F-C:8, F-D:2, F-E:15.

b) ABEGH, ABGEH, BAEGH, BAGEH, BEAGH, BEGAH, BEGHA, BGAEH, BGEAH, BGEHA.

Uppgift 6 (8 p)

De tre första talen 44, 8 resp. 32 sätts in på platserna 8, 0 resp. 5:

0	**8**
1	13
2	
3	
4	40
5	**32**
6	
7	
8	**44**
9	57
10	
11	

Tabellen är nu mer än halvfull och dess storlek är inte ett primtal. Då man försöker sätta in 12 terminerar inte sökningen efter en ledig plats.

Uppgift 7 (10 p)

```
public static void printLevelOrder(TreeNode t) {
    if ( t == null )
        return;
    else {
        Queue<TreeNode> q = new LinkedList<TreeNode>();
        q.add(t);
        while ( ! q.isEmpty() ) {
            TreeNode n = q.remove();
            System.out.print(n.element + " ");
            if ( n.left != null )
                q.add(n.left);
            if ( n.right != null )
                q.add(n.right);
        }
    }
}
```