Lösningsförslag till tentamen 170818

Uppgift 1

a) Sant.

b) Sant.

c) Sant.

d) Falskt. IPv6 anger en IP-adress med 132 bitar.

e) Falskt. *Spoofing* åsyftar användning av förfalskad eller lånad identitet på Internet, ofta för bedrägligt syfte.

f) Falskt! Tidskomplexiteten är $\Theta(n^2 \log n)$.

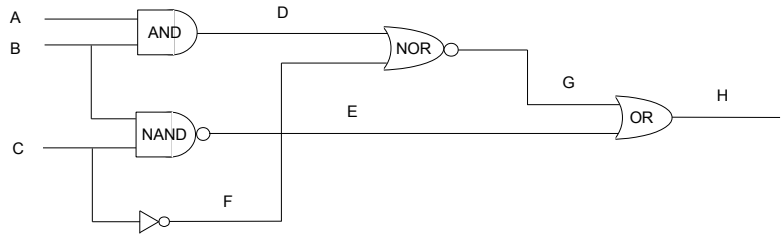
g) Sant.

h) Sant

i) Falskt. Man syftar på att utvecklingen inom mjukvaruindustrin går allt snabbare och nya produkter måste lanseras allt snabbare för att inte bli utkonkurrerad.

j) Falskt. Har modulerna hög coupling och låg cohesion är programsystemet svårt att bygga ut och modifiera, beroende på att en förändring i en moduler påverkar andra moduler.

Uppgift 2



Uppgift 3

Ett fotografi är en avbildning av objekt i verkligheten som skapas genom att ljus fångats på ett ljuskänsligt medium. För att en datorgenererad bild skall bli realistisk gäller det att återskapa hur energin från ljuskällor reflekteras av olika ytor i bilden.

Detta kan till exempel vara metoder för att förse en bild med reflektioner, skuggor och mer avancerad belysningsteknik. Skuggor och reflektioner är mycket viktiga i många sammanhang, eftersom människan använder sig av dessa för att avgöra var olika objekt befinner sig i förhållande till varandra.

Uppgift 4

Θ-notationen används för att beskriva en algoritms tidsåtgång utan att ta hänsyn till detaljer och utan att ange tidsåtgången i absoluta mått (vilket naturligtvis är beroende av hur snabb hårdvaran är). Det som beskrivs är hur tidsåtgången växer beroende på storleken av den datamängd som algoritmen bearbetar. Man tar endast hänsyn till den "dominerande termen", dvs endast den del i algoritmen där den största tidsåtgången förbrukas.

Att två algoritmer tillhör samma Θ-klass innebär att båda algoritmerna har samma tillväxthastigheten på tidsåtgången. En algoritm som har tidskomplixiteten $\Theta(n^2)$ har således en tillväxt på tidsåtgången som är kvadratisk med hänseende till den bearbetade datamängdens storlek.

Eftersom Θ-notationen inte anger någon absolut tidsåtgång, kan en algoritm som tillhör en Θ-klass med en snabb tillväxtshastighet i absoluta mått kräva mindre tidsåtgång för små datamängder än en algoritm som tillhör en Θ-klass med en långsammare tillväxtshastighet.

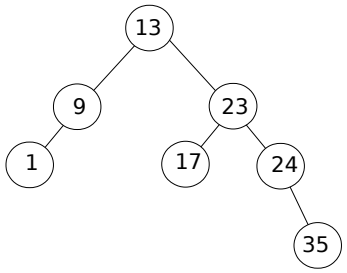
Uppgift 5

Ett *binärt sökträd* är ett *binärt träd*, där varje nod innehåller ett värde från en ordnad mängd. För varje nod n i ett binärt sökträd gäller följande:

- samtliga noder i det vänstra subträdet till *n* innehåller värden som är mindre än värdet i noden *n*.

- samtliga noder i det högra subträdet till *n* innehåller värden som är större än värdet i noden *n*.

Betrakta nedanstående binära sökträd:



Sökning efter ett element t.ex. det som har värdet 17 tillgår enligt följande:

17 är större än värdet 13 som finns i roten, därför fortsätter sökningen i det högra subträdet.

17 är mindre än värdet 23 som värdet i roten av det subträd som vi nu genomsöker, därför fortsätter sökningen i det vänstra subträdet till detta subträd.

17 är lika med värdet 17 som är värdet i det subträd som vi nu genomsöker. Alltså har gvi lokaliserat det eftersökta elementet.

Om vi söker efter ett element som inte finns i trädet kommer när det subträd som skall genomsökas är tomt.

Uppgift 6

Namn	**Saldo**
Jens Andersson	25090
Stefan Lilja	65435
Per Persson	34550

Uppgift 7

a) Utskriften blir

[6, 6, 6, 10, 11]

b) Utskriften blir:

8
27
64
125
125
125
125
125

Uppgift 8

```
import javax.swing.*;
public class Wind {
    public static void main(String[] args){
        boolean done = false;
        while (!done) {
            String indata = JOptionPane.showInputDialog("Ange vindstyrkan i m/s:");
            if (indata ==  null)
                done = true;
            else {
                double wind = Double.parseDouble(indata);
                if ( wind < 0 || wind > 115)
                    JOptionPane.showMessageDialog( null, "Felaktig indata! Försök igen!");
                else {
                    String name = term(wind);
                    JOptionPane.showMessageDialog( null, "En vind på " + wind + " m/s kallas på sjön för "  + name);
                }
            }
        }
    } // main

    public static String term( double wind){
        String name;
        if (wind <= 0.2)
            name = "stiltje";
        else if (wind <= 13.9)
            name = "bris";
        else if (wind <= 24.5)
            name = "kuling";
        else if (wind <= 32.7)
            name = "storm";
        else
            name = "orkan";
        return name;
    }//term
} //Wind
```

Uppgift 9

a)

```
public class MexicoDice {
    private Die d1;
    private Die d2;
    private int value;

    public  MexicoDice() {
        Die d1 = new Die();
        Die d2 = new Die();
        value = 0;
    }//constructor

    public void roll() {
        d1.roll();
        d2.roll();
        int r1 = d1.getValue();
        int r2 = d2.getValue();
        if (r1 >= r2) {
            value = 10 * r1 + r2;
        } else {
            value = 10 * r2 + r1;
        }
    }//roll

    public int getValue() {
        return value;
    }//getValue
}//MexicoDice
```

b)

```
public class Util {
    public static int lowestScoreValue(int score1,  int score2) {
        if (toSerialNumber(score1)< toSerialNumber(score2))
            return score1;
        else
            return score2;
    }//lowestScoreValue

    private static int toSerialNumber(int score) {
        if (score == 21)
            return 1000;
        if (score / 10 == score % 10)
            return 10 * score;
        return score;
    }//toSerialNumber
}//Util
```

a)

```
public class MexicoDice {
    private Die d1;
    private Die d2;
    private int value;
    public  MexicoDice() {
        Die d1 = new Die();
        Die d2 = new Die();
        value = 0;
    }//constructor
    public void roll() {
        d1.roll();
        d2.roll();
        int r1 = d1.getValue();
        int r2 = d2.getValue();
        if (r1 >= r2) {
            value = 10 * r1 + r2;
        } else {
            value = 10 * r2 + r1;
        }
    }//roll
    public int getValue() {
        return value;
    }//getValue
}//MexicoDice
```

b)

```
public static int lowestScoreValue(int score1,  int score2) {
    if (toGrade(score1)< toGrade(score2))
        return score1;
    else
        return score2;
}//lowestScoreValue

public static int toGrade(int score) {
    if (score == 21)
        return 1000;
    if (score/10 == score%10)
        return 10 * score;
    return score;
}//toGrade
```

Uppgift 10

a)

```
public static int secondLargest(int[] a) {
    if (a == null || a.length < 2)
        throw new IllegalArgumentException();
    int max;
    int secMax;
    if (a[0] > a[1]) {
        max = a[0];
        secMax = a[1];
    } else {
        max = a[1];
        secMax = a[0];
    }
    for (int i = 2; i < a.length; i++) {
        if (a[i] > max) {
            secMax = max;
            max = a[i];
        } else if (secMax < a[i] && a[i] < max) {
            secMax = a[i];
        }
    }
    return secMax;
}//secondLargest
```

b)

```
public static int[][][] frosty(int[][][] samples) {
    int[][][] newSamples = new int[samples.length][samples[0].length][3];
    for (int row = 0; row < samples.length; row = row + 1) {
        for (int col = 0; col < samples[row].length; col = col + 1) {
            int xn = disturb(row, samples.length-1);
            int yn = disturb(col, samples[0].length-1);
            for (int c = 0; c < 3; c++) {
                newSamples[row][col][c] = samples[xn][yn][c];
            }
        }
    }
    return newSamples;
}//frosty

private static int disturb(int v, int max) {
    java.util.Random r = new java.util.Random();
    int nx;
    do {
        nx = v +  r.nextInt(2 * 5 + 1) - 5;
    } while (nx < 0 || nx > max);
    return nx;
}
```

Uppgift 11

a)

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class SimpleEggTimer extends JPanel implements ActionListener {
    private JButton plus = new JButton("+");
    private JButton start = new JButton("Start");
    private JLabel talLabel;
    private Timer timer = new Timer(1000, this);
    private int nrOfSeconds = 0;

    public SimpleEggTimer() {
        talLabel = new JLabel("0", JLabel.CENTER);
        setLayout(new BorderLayout());
        add("Center", talLabel);
        add("East", plus);
        add("South", start);
        plus.setBackground(Color.yellow);
        talLabel.setBackground(Color.white);
        talLabel.setOpaque(true);
        plus.addActionListener(this);
        start.addActionListener(this);
        start.setEnabled(false);
        Font font = new Font("Times", Font.PLAIN, 36);
        plus.setFont(font);
        talLabel.setFont(font);
    } //konstruktor

    public void actionPerformed(ActionEvent e) {
        if (e.getSource() == timer)
            countDown();
        else if (e.getSource() == plus)
            setTime();
        else if (e.getSource() == start)
            startClock();
    }//actionPerformed

    private void countDown() {
        nrOfSeconds = nrOfSeconds -1;
        talLabel.setText(" " + nrOfSeconds);
        if (nrOfSeconds != 0) {
            Toolkit.getDefaultToolkit().beep();
            timer.stop();
            plus.setEnabled(true);
        }
    } // countDown

    private void setTime() {
        nrOfSeconds = nrOfSeconds + 1;
        talLabel.setText(" " + nrOfSeconds);
        start.setEnabled(true);
    } //setTime

    private void startClock() {
        timer.start();
        start.setEnabled(false);
        plus.setEnabled(false);
    } //startClock
}//SimpleEggTimer
```

b)

```
import javax.swing.*;
public class ShowSimpleEggTimer extends JFrame {
    public ShowSimpleEggTimer() {
        SimpleEggTimer clock  = new SimpleEggTimer();
        add(clock);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(300, 150);
        setVisible(true);
    } //konstruktor

    public static void main(String[] arg) {
        JFrame w = new ShowSimpleEggTimer();
    }//main
}//ShowSimpleEggTimer
```