Lösningsförslag till tentamen 160819

Uppgift 1

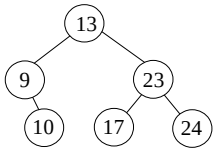
a) Sant.

b) Falskt. Applikationslagret skickar till transportlagret.

c) Falskt. Det är en gateway som kopplar ihop två nätverk som använder olika protokoll. En router kopplar samman och dirigerar datatrafiken mellan nätverk som har samma protokoll.

d) Sant.

e) Falskt. Minsta värdet I ett binärt sökträd finns i *noden längst till vänster*, men denna behöver inte vara ett löv. Nedanstående binära sökträd är ett exempel på detta



f) Falskt. Open-source development innebär att källkoden för en programvara är öppen för vem som helst att ändra och anpassa.

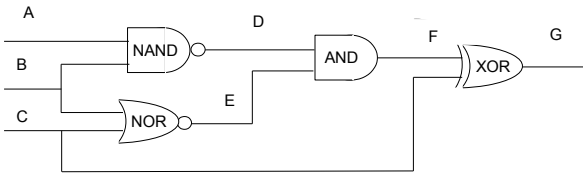
g) Sant.

h) Falskt. Tendensen är att ha mindre team för att minska onödig byråkrati och därigenom öka produktionstakten.

i) Sant.

j) Falskt. Att skapa mjuka skuggor kräver mycket mer beräkningskapacitet.

Uppgift 2



A	B	C	D	E	F	G
0	0	0	1	1	1	1
0	0	1	1	0	0	1
0	1	0	1	0	0	0
0	1	1	1	0	0	1
1	0	0	1	0	0	0
1	0	1	1	1	1	1
1	1	0	0	0	0	0
1	1	1	0	0	0	1

Uppgift 3

Moore's lag innebär att antalet transistorer som ryms på ett chip fördubblas var 18:e månad. Det är denna utveckling som gjort att processorkraften ökat explosionsartat samtidigt som de fysiska enheterna minskat I storlek. Tack vare Moore's lag är dagens mobiltelefoner 1000 gånger snabbare än de stordatorer som användes i NASAs Apollo-program. Idag har majoriteten av alla svenskar en smartphone med avancerade program som de använder dagligen. Detta kulturella paradigmskifte skulle vara omöjligt utan Moore's lag.

Uppgift 4

Trojaner: Skadlig kod som gömmer sig inuti ett annat program. En trojan kan till exempel spionera på användaren, göra betalningar i användarens namn, skicka skräppost eller radera filer.

Datormask: Datormaskar är bland de vanligaste av skadlig programvara, och sprider sig genom att utnyttja sårbarheten i operativsystemet. Datormaskar skadar vanligtvis användaren genom att konsumera bandbredd och att överladda webbservrar. Datormaskar kan även innehålla och exekvera kod som utför ytterligare processer utöver att sprida masken vidare. Skillnaden mot ett virus är att datormaskar har möjligheten att självreplikera och sprida sig vidare, medan virus kräver mänsklig interaktion för att spridas.

Rootkit: Ett rootik är skadlig programvara designad för att på håll ta över och kontrollera en dator utan att bli upptäckt. När ett rootkit har blivit installerat är det möjligt för den att stjäla filer, exekvera kod, modifiera mjukvara och inställningar, kontrollera datorn för att tillsammans med andra datorer överbelasta nätverk, eller andra skadliga processer.

Spyware: Mjukvara som spionerar på användaren utan dess tillåtelse. Programmet kan t ex registrera vilka knapptryck som användaren trycker för att luska fram lösenord, eller snoka upp data, likt bankkontoinformation.

Uppgift 5

a) Tidskomplexiteten är $\Theta(n^2)$.

b) Tidskomplexiteten är $\Theta(n^2)$.

Uppgift 6

Name	Quantity
Halo 3	4
Farmville	8
Mass Effect	2
Oblivion	7

Uppgift 7

a) Utskriften blir:

[6, 3, 3, 5, 7]

b) Felet beror på att att metoden biggest är en instansmetod, men att anropet biggest anropas i main-metoden som om den vore en klassmetod.

Det enklaste och naturligaste sättet att korrigera felet är att göra om metoden biggest till en klassmetod:

```
private static int  biggest( int [] vekt) {
    int  max = vekt[0];
    for  (int  i = 1; i < vekt.length; i = i + 1) {
        if  (vekt[i] > max)
            max = vekt[i];
    }
    return  max;
}
```

Ett alternativt sätt att korrigera felet (som är onaturligare) är att låta metoden biggest stå kvar som en instansmetod och skapa en instans i main och anropa metoden för denna instans:

```
public static void  main (String[] args) {
    int [] v = {4, 7, 3, 11, 7, 9};
    Uppgift7B obj = new  Uppgift7B();
    System.out.println("Det största talet i fältet är " + obj.biggest(v));
}
```

c) Utskriften blir:

x = 1, y = 10
x = 8, y = 10

d) Utskriften blir:

wonderful is life

Metoden mystery tar en sträng sentence och returnerar en ny sträng, vilken innehåller *orden* som finns i sentence i omvänd ordning.

Uppgift 8

```
import java.util.*;
public class Student {
    private String name;
    private String programme;
    private  ArrayList<Course> passed;
    public Student(String name, String programme) {
        this.name = name;
        this.programme = programme;
        passed = new ArrayList<Course>();
    }//constructor
    public String getName() {
        return name;
    }//getName
    public String getProgramme() {
        return programme;
    } //getProgramme
    public void addCourse(Course c) {
        passed.add(c);
    }//add
    public int getNrPassed() {
        return passed.size();
    }//getNrPassed

    public boolean hasPassed(String code) {
        for (Course c : passed )
            if (c.getCode().equals(code))
                return true;
        return false;
    }//hasCourse
    public double getTotalCredits() {
        double res = 0;
        for (Course c : passed )
            res = res + c.getCredits();
        return res;
    }//getTotalCredits

    public String toString() {
        String out = name + ", " + programme + "\n";
        for (Course c : passed )
            out = out + c.toString() + "\n";
        return out;
    }//toString
}//Student
```

```
//Förhindra att dubletter sparar
public void addCourse(Course c) {
        if (!passed..contains(c))
            passed.add(c);
    }//add
```

Alternativa implementationer:

```
public boolean hasPassed(String code) {
    for (int i = 0; i < passed.size();  i = i + 1 )
        if (passed.get(i).getCode().equals(code))
            return true;
    return false;
}//hasCourse

public double getTotalCredits() {
    double res = 0;
    for (int i = 0; i < passed.size(); i++)
        res = res + passed.get(i).getCredits();
    return res;
}//getTotalCredits

public String toString() {
    String out = name + ", " + programme + "\n";
    for (int i = 0; i < passed.size();  i = i + 1)
        out = out + passed.get(i).toString() + "\n";
    return out;
}//toString
```

Uppgift 9

```
import javax.swing.*;
import java.util.*;
import java.text.*;
public class Calories {
    public static void main(String[] arg) {
        while(true) {
            String indata = JOptionPane.showInputDialog("Antalet kalorier och antalet gram fett: ");
            if (indata == null)
                break;
            Scanner sc = new Scanner(indata);
            double nrCalories  = sc.nextDouble();
            double gramsOfFat  = sc.nextDouble();
            if (nrCalories < 0 || gramsOfFat < 0) {
                JOptionPane.showMessageDialog(null,"Negativa värden kan får ej ges!");
            }
            else {
                double caloriesFromFat = getCalories(gramsOfFat);
                double precentageFromFat =  caloriesFromFat/ nrCalories *100;
                if (precentageFromFat > 100)
                    JOptionPane.showMessageDialog(null,"Felaktig indata!\n" +
                                                       "Andelen fett kan inte överskrida 100%!");
                else if (precentageFromFat <= 30)
                    JOptionPane.showMessageDialog(null, String.format("%.2f", precentageFromFat)  +
                                                        " procent av kalorierna kommer från fett. \n" +
                                                        " Kategori: GRÖN " );
                else if (precentageFromFat <= 40)
                    JOptionPane.showMessageDialog(null, String.format("%.2f", precentageFromFat)  +
                                                        " procent av kalorierna kommer från fett.\n" +
                                                        " Kategori: GULT " );
                else
                    JOptionPane.showMessageDialog(null, String.format("%.2f", precentageFromFat)  +
                                                       " procent av kalorierna kommer från fett.\n"  +
                                                       " Kategori: RÖTT " );
            }
        }//while
    }//main

    public static double getCalories(double grams) {
        return 7.7*grams;
    }// getCalories
}// Calories
```

Uppgift 10

a)

```
public static void increaseAverage(int[] a, int[] b) {
    if (a == null || b == null ||  a.length == 0 || b.length == 0 )
            throw new IllegalArgumentException();
    boolean success = false;
    for (int i = 0; i < a.length; i++) {
        if (a[i] < getMean(a) && a[i] > getMean(b)) {
            System.out.println("Move " + a[i] + " from A to B");
            success = true;
        }
     }
     for (int i = 0; i < b.length; i++) {
        if (b[i] < getMean(b) && b[i] > getMean(a)) {
           System.out.println("Move " + b[i] + " from B to A");
           success = true;
        }
     }
     if (!success) {
        System.out.println("Impossible to increase average in both groups!");
     }
   }//advice

private static double getMean(int[] arr) {
    double sum = 0;
    for (int i : arr) {
        sum = sum + i;
    }
    return sum / arr.length;
}//getMean
```

b)

```
public static int[][][] withFrame(int[][][] samples, int[] color) {
    int frameSize = samples[0].length / 10;
    int[][][] newSamples = new int[samples.length + 2*frameSize][samples[0].length + 2*frameSize][3];

    //Kopiera den orginalbilden till den nya bilden
    for (int row = 0; row < samples.length; row = row + 1) {
        for (int col = 0; col< samples[row].length; col = col + 1) {
            for (int c = 0; c < 3; c = c + 1) {
                newSamples[row+frameSize][col + frameSize][c] = samples[row][col][c];
            }
        }
    }

    //Skapa övre och under ram
    for (int row = 0; row < frameSize; row = row + 1) {
        for (int col = 0; col < newSamples[row].length; col = col + 1) {
            for (int c = 0; c < 3; c = c + 1) {
                newSamples[row][col][c] = color[c];
                newSamples[newSamples.length-1 - row][col][c] = color[c];
            }
        }
    }

    //Skapa vänstra och högra ram
    for (int row = frameSize; row < newSamples.length - frameSize; row = row + 1) {
        for (int col = 0; col < frameSize; col = col + 1) {
            for (int c = 0; c < 3; c = c + 1) {
                newSamples[row][col][c] = color[c];
                newSamples[row][newSamples[row].length -1 -col][c] = color[c];
            }
        }
    }
     return newSamples;
}//withFrame
```

Alternativ lösning:

```
public static int[][][] withFrame(int[][][] samples, int[] color) {
    int frameSize = (int) (0.1*samples[0].length);
    int[][][] newSamples = new int[samples.length + frameSize][samples[0].length + 2*frameSize][3];
    for (int row = 0; row < newSamples.length; row = row + 1) {
        for (int col = 0; col< newSamples[row].length; col = col + 1) {
            for (int c = 0; c < 3; c = c + 1) {
                if (row < frameSize || row >= samples.length +  frameSize
                   || col < frameSize || col >= samples[0].length +  frameSize) {
                    newSamples[row][col][c] = color[c];
                }
                else {
                    newSamples[row][col][c] = samples[row - frameSize][col-frameSize][c];
                }
            }
        }
    }
    return newSamples;
}//withFrame
```

Uppgift 11

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class CodeLock extends JFrame implements ActionListener {
    private NumberButton[] t = new NumberButton[5];
    private int theCode = 0;
    public CodeLock(int theCode) {
        if (theCode < 0 || theCode > 99999)
            throw new IllegalArgumentException();
        this.theCode = theCode;
        Font font = new Font("Times", Font.PLAIN, 36);
        setLayout(new GridLayout(1, 5,3,3));
         for (int i = 0; i < t.length; i = i + 1) {
            t[i] = new NumberButton(0);
            add(t[i]);
            t[i].addActionListener(this);
            t[i].setBackground(Color.YELLOW);
            t[i].setOpaque(true);
            t[i].setFont(font);
        }
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setSize(320, 100);
        setVisible(true);
    }//konstruktor

    public void actionPerformed(ActionEvent e) {
        for (int i = 0; i < t.length; i = i + 1) {
            if (e.getSource() == t[i]) {
                t[i].add();
            }
        }
        correctCodeGiven();
    }// actionPerformed

    private void correctCodeGiven() {
        int givenCode = 0;
        for (int i = 0; i < t.length; i = i + 1) {
            givenCode = 10 *givenCode + t[i].getNumber();
        }
        if (givenCode == theCode) {
            for (int i = 0; i < t.length; i = i + 1)
                t[i].setEnabled(false);
        }
    }//correctCodeGiven
}// CodeLock


public class Main {
    public static void main (String[] arg) {
        CodeLock k = new CodeLock(12221);
    } //main
}// Main
```