Lösningsförslag till tentamen 160315

Uppgift 1

a) Falskt. $8A_{16} = 138_{10} = 10001010_2$

b) Falskt. Enligt *Moores lag* fördubblas antal transistorer som ryms på en given yta var 18:e månad.

c) Sant. Sant. UDP är ett enklare protokoll än TCP. UDP har t.ex. inga funktioner för att rätta överföringsfel, TCP har. Men om man tittar på en film behöver inte varje filmruta vara helt perfekt. Hastigheten, att filmen flyter på, är viktigare en kvalitén.

d) Falskt. Mellan transportlagret och länklagret finns nätverkslagret (*network layer*).

e) Sant.

f) Falskt. En glupsk algoritm ger sällan den optimala lösningen, men oftast en lösning som ligger nära den oprtimala.

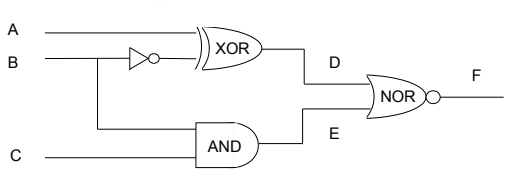
g) Falskt. Man eftersträvar att eliminera all redudans.

h) Sant.

i) Sant.

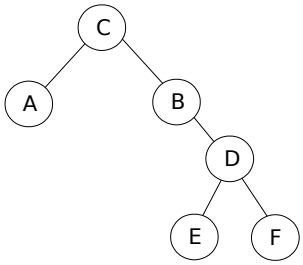
j) Sant.

Uppgift 2



A	**B**	**¬B**	**C**	**D**	**E**	**F**
0	0	1	0	1	**0**	**0**
0	0	1	1	1	**0**	**0**
0	1	0	0	0	**0**	**1**
0	1	0	1	0	**1**	**0**
1	0	1	0	0	**0**	**1**
1	0	1	1	0	**0**	**1**
1	1	0	0	1	**0**	**0**
1	1	0	1	1	**1**	**0**

Uppgift 3



Uppgift 4

Förklarningen är datatypen **int** lagras på binärform i 32 bits enligt 2-komplementsform. När 2-komplementsfrom nyttjas är den mest signifikanta biten en teckenbit. Teckenbiten 0 anger ett positivt tal och 1 anger ett negativt tal. Talet 2147483647_{10} lagras alltså på binärform i 32bitar enligt

```
0111. . . 1111
```

Om vi nu addera 1 till detta värde får vi

```
0111. . . 1111
          +1
1000. . . 0000
```

Den mest signifikanta siffran i det erhållna värdet är 1, vafrör vi har ett negativt tal. Vid 2-komplementsform erhålls bitmönstret för en negativt tal enligt:

1. Beräkna bitmönstret för motsvarande positiva heltal
2. Invertera detta bitmönster (dvs byt alla 0:or mot 1:or och vice versa).
3. Addera 1 till det bitmönster som erhölls i steg 2.

För att erhålla absolutbeloppet av det negativa talet

```
1000. . . 0000
```

skall vi göra tvärsom, dvs

1. Subtrahera 1 från bitmönstret
2. Invertera det erhållna bitmönstet (dvs byt alla 0:or mot 1:or och vice versa).

Detta resulterar i

```
1. 1000. . . 0000
      -1
   0111. . . 1111
2. 1000. . . 0000
```

Decimalt är detta värde 2^{31} = 2147483648. Därav utskriften – 2147483648.

Uppgift 5

En risk är att man kan bli utsatt för sabotage och vandalism av olika slag, t.ex. *virus*, *maskar* och *trojanska hästar*. När nya sådana börjar florera utveckals snabbt olika *virusskydd*. Således skall man installera virusskydd och uppdatera dessa ofta. Vanligtvis sprids virus via email, varför man bör installera spamfilter samt vara försiktig med att öppna filer som bifogas i email från okända avsändare.

En annan risk är att råka ut för *informationsstöld*, via sniffing (avlyssning) eller phishing (förfrågan). Det finns olika typer av åtgärder som kan vidtas. För att förhindra avlyssning kan kryptering användas. Givitvis skall man aldrig besvara ett email som efterfrågar känslig och privat information, såsom lösenord och kontonummer.

En skyddsmekanism mot både sabotage och informationsstöld är att installera en *brandvägg*.

Uppgift 6

Följande tabell erhålls:

Maträtt	**Pris**
Kokt ishavstorsk med hackat ägg	139
Omelett	89
Stekt sill	79

Uppgift 7

a) Utskriften blir:

45992

Metoden tar en strängrepresentation av ett icke-negativt hexadecimalt heltal och översätter detta till ett decimalt heltal.

b) Utskriften blir:

4 2
6 8

c) Om parametern number har ett värde som är mindre eller lika med 0 returnerar metoden ett felaktigt värde (nämligen 0). En korrekt implementation har utseendet:

```
public static int nrOfDigits(int number) {
    if (number == 0)
        return 1;
    number = Math.abs(number);
    int digits = 0;
    while (number > 0) {
        digits =digits + 1;
        number = number / 10;
    }
    return digits;
}//nrOfDigits
```

Uppgift 8

```
import javax.swing.JOptionPane;
import java.util.Scanner;
public class Stock {
    public static void main(String args[]) {
        boolean done = false;
        while (!done) {
            String indata = JOptionPane.showInputDialog("Ge antal aktier, inköpspris, provision vid inköp, \n"
                                                        +  "försäljningspris samt provision vid försäljning");
            if (indata == null)
                done = true;
            else {
                Scanner sc = new Scanner(indata);
                int NS = sc.nextInt();
                double PP = sc.nextDouble();
                double PC = sc.nextDouble();
                double SP = sc.nextDouble();
                double SC = sc.nextDouble();
                if (NS < 0 || PP < 0 || PC < 0 || SP < 0 || SC < 0)
                    JOptionPane.showMessageDialog(null, "Otillåten indata!");
                else {
                    double profit = profit(NS, PP, PC, SP, SC);
                    String output;
                    if (profit < 0) {
                        output = String.format("En förlust på %.2f kronor!", Math.abs(profit));
                    }
                    else {
                            output = String.format("En vinst på %.2f kronor!", profit);
                    }
                    JOptionPane.showMessageDialog(null, output);
                }
            }
        }
    }//main

    public static double profit(int NS, double PP, double PC, double SP, double SC)  {
            return (NS * SP - SC) - (NS * PP + PC);
    }//profit
}//Stock
```

Uppgift 10

a)

```
import  java.awt.Point;
public class  City {
    private  String name;
    private  Point position;
    private boolean  visited;
    public City(String name, Point position) {
        this.name = name;
        this.position = position;
        visited = false ;
    }
    public void  setVisited() {
        visited = true ;
    }
    public boolean  getVisited() {
        return visited;
    }
    public String getName() {
        return name;
    }
    public double distanceTo(City other) {
        return  Math.sqrt(Math.pow((this.position.getX() - other.position.getX()), 2)
                        + Math.pow((this.position.getY() - other.position.getY()), 2));
     }
}//City
```

b)

```
public static City getClosesCiy(ArrayList<City> cities, City presentCity) {
    double  minDist = Double.MAX_VALUE;
    City  cand = cities.get(0);
    for  (City c : cities) {
        double  dist = presentCity.distance(c);
        if  (dist< minDist && !c .getVisited()) {
            minDist = dist;
            cand = c;
        }
    }
    return  cand;
} //getClosesCiy
```

c)

```
public static void  calculateRoute(ArrayList<City> citiesToVisit, City startCity) {
    startCity.setVisited();
    System.out.print("Resvägen är: " + startCity.getName());
    City presentCity = startCity;
    double  totalDistance = 0;
    for  (int i = 1; i < citiesToVisit.size(); i = i + 1) {
        City nextCity = getClosesCiy(presentCity, citiesToVisit);
        nextCity.setVisited();
        totalDistance = totalDistance + presentCity.distance(nextCity);
        System.out.print(" - "+ nextCity.getName());
        presentCity = nextCity;
    }
    System.out.println(" - " + startCity.getName());
    totalDistance = totalDistance + presentCity.distance(startCity);
    System.out.println("Totala ressträckan är: " + totalDistance);
} // calculateRoute
```

Uppgift 10

```
public static double geometricMean(int[] values) {
    if (values == null)
        throw new NullPointerException();
    if (values.length == 0)
        throw new IllegalArgumentException();
    double prod = 1;
    for (int i = 0; i < values.length; i++) {
        if (values[i] <= 0)
            throw new NumberFormatException();
        prod = prod * values[i];
    }
    return Math.pow(prod, 1.0 / values.length);
}//geometricMean
```

Uppgift 11

```
public static int[][][] decodeGrayPicture(int[][][] colourPicture) {
    int[][]grayPicture = new int[colourPicture .length][colourPicture[0].length];
    for (int row = 0; row < grayPicture.length; row = row + 1) {
        for (int col = 0; col < grayPicture[row].length; col = col + 1) {
            grayPicture[row][col] = decodePixel(colourPicture[row][col]);
        }
    }
    return grayPicture;
} //decodeGrayPicture

private int decodePixel(int[][][] colourPixel) {
    int r = colourPixel[0]% 10;
    int g = colourPixel[1] % 10;
    int b = colourPixel[2] % 10;
    return 100*r + 10*g + b;
}//decodePixel
```

Alternativ:

```
public static int[][][] decodeGrayPicture(int[][][] colourPicture) {
    int[][]grayPicture = new int[colourPicture .length][colourPicture[0].length];
    for (int row = 0; row < grayPicture.length; row = row + 1) {
        for (int col = 0; col < grayPicture[row].length; col = col + 1) {
            int r = colourPicture[row][col][0]% 10;
            int g = colourPicture[row][col][1] % 10;
            int b = colourPicture[row][col][2] % 10;
            grayPicture[row][col] = 100*r + 10*g + b;
        }
    }
    return grayPicture;
} //decodeGrayPicture
```

Uppgift 12

a)

```
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;
public class GrowingCircle extends JPanel  implements ActionListener {
private javax.swing.Timer t = new javax.swing.Timer(500, this);
    private int radie = 0;
    public GrowingCircle() {
        setBackground(Color.YELLOW);
        radie = 1;
        t.start();
    }//construktor

    public void paintComponent(Graphics pen) {
        super.paintComponent(pen);
        pen.setColor(Color.PINK);
        int x = getWidth()/2 - radie;
        int y = getHeight()/2 - radie;
        int b = 2*radie;
        int h = 2*radie;
        pen.fillOval(x, y, b, h);
    }

    public void actionPerformed(ActionEvent e) {
        if (radie < getHeight()/2 && radie < getWidth()/2) {
            radie = radie +1;
            repaint();
        } else {
            t.stop();
        }
    }//actionPerformed
}//GrowingCircle
```

b)

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class ShowGrowingCircle extends JFrame {

    public ShowGrowingCircle() {
        add(new GrowingCircle());
        setDefaultCloseOperation(EXIT_ON_CLOSE);
    }//construktor

    public static void main(String[] args) {
        ShowGrowingCircle c = new ShowGrowingCircle();
        c.setSize(325,225);
        c.setVisible(true);
    }//main
}//  ShowCircle
```