Lösningsförslag till tentamen 170609

Uppgift 1

a) Falskt! Bitmönstret är 10010111

b) Falskt! Memory Manager handhar

c) Falskt. En *trojansk häst* är ett program i förklädnad. Det är således program som på ytan ser ut att vara ett vanligt program t.ex ett spel, men som när det startas även gör annat i avsikt att skada den angripna datorn, som t.ex att radera filer eller ändra inställningar.

d) Falskt. Det största värdet finns i noden längst till höger i trädet.

e) Sant

f) Sant

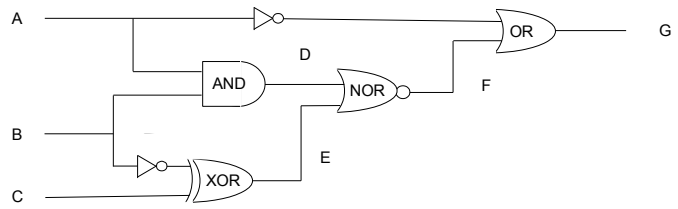
g) Falskt! Den vanligaste typen av databaser är relationsdatabaser.

h) Sant

i) Falskt. Man eftersträvar *låg coupling* och *hög cohesion*.

j) Falskt! 3D-rendering innebär att ett 3D-objekt projiceras till ett 2D-objekt.

Uppgift 2



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

Uppgift 3

$\Theta(1)$, $\Theta(^2\log n)$, $\Theta(n)$, $\Theta(n\ ^2\log n)$, $\Theta(n^2)$, $\Theta(n^3)$

Uppgift 4

Vid binärsökning måste datamängden som skall sökas igenom vara *sorterad*. Sökningen utförs i flera steg och i varje steg halveras antalet element i den kvarvarande datamängden som behöver genomsökas.

Antag att vi har följande datamängd

Anne Beda Christina Doris Eva Fiona Gunnel Helen Irene Jane Klara Laura Moa Nina Olga

och att vi söker efter Beda.

Sökningen börjar i mittelementet i den ursprungliga sökmängden, dvs med elementet Helen. Eftersom nu Helen är större än Beda, vet vi att Beda måste finnas tidigare i datamändgen än Helen. Således kan vi koncentrera sökningen till datamängden

Anne Beda Christina Doris Eva Fiona Gunnel

Nu återupprepas samma förfarande, dvs vi undersöker mittelemetet, vilket är Doris. Eftersom Doris är större än Beda kan den fortsatta sökningen koncentreras till datamängden

Anne Beda Christina

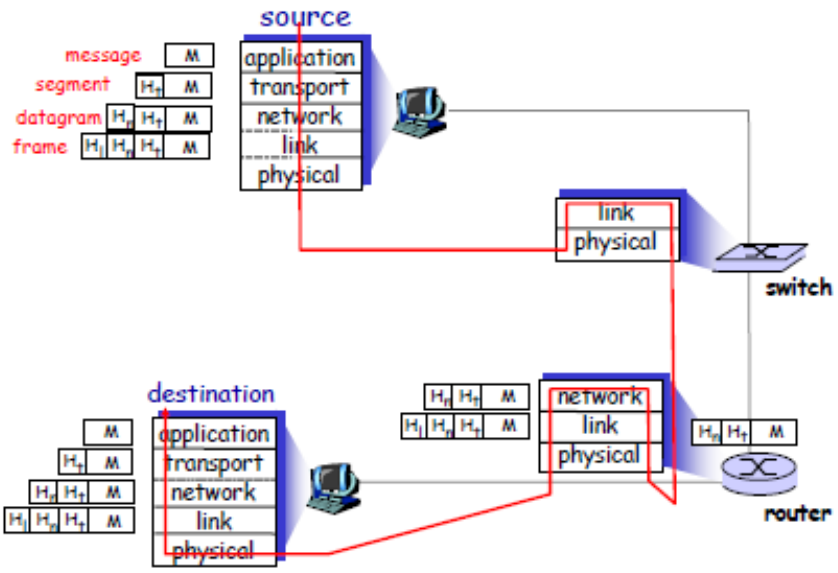
Förfarandet återupprepas, och vi undersöker mittelementet, vilket är Beda. Vi har hittat vad vi söker!

Eftersom sökmängden, om vi inte tidigare hittar vad vi söker, reduceras ned till ett element kommer vi att hitta vad vi söker eller konstatera att det vi söket inte finns i den ursprungliga datamängden. Binärsökning har tidskomplexiteten $\Theta(^2\log n)$.

Uppgift 5

Internethierarkin är uppbyggd av fyra lager:

Applikationslagret - Transportlagret - Nätverkslagret - Länklagret



Hierarkin går uppifrån och ner då information skickas från sändaren och nerifrån och upp då den mottas av mottagaren.

Applikationslagret: Applikationsprogram som använder sig av Internet, exempelvis webbläsare och chatprogram, interagerar direkt med varandra genom applikationslagret. Applikationslagret förbereder meddelandet och märker det med en slutadress.

Transportlagret: Delar upp meddelandet i mindre delar, så kallade paket(*packets*). Dessa paket skickas sedan i en angiven ordning. Omvänt så samlar transport ihop paketen och sätter ihop dessa till det ursprungliga meddelandet

Nätverkslagret: Nätverkslagret adresserar meddelanden och översätter logiska adresser och namn till fysiska adresser. Det avgör också vilken väg i nätverket som meddelanden skall färdas och hanterar trafiksproblem som stockningar.

Länklagret: Ansvarar för att överföra data mellan två närliggande noder i nätet genom att använda sig av det fysiska lagrer.

Uppgift 6

Model	Price	Year	To100
Z4 sDrive35i	420000	2016	5.2
Carrera 911	360000	2004	4.7
Boxster S	280000	2012	5.0

Uppgift 7

a) Utskriften blir

[2, 4, 7, 8, 12]

b) Utskriften blir

Vad blir 4*4? 30
Vad blir 5*6? 30

c) Utskriften blir:

MELBORP AGNÅM SNNIF TED

d) För att kunna exekvera en klass måste klassen ha en statisk main-metod, som är specificerad enligt:

public static void main(String [] args)

Klassen Wrong har ingen sådan metod! Istället finns en main-metod som är en instansmetod. Denna måste alltså göras om till en klassmetod enligt:

```
public class Wrong {
public static void main(String [] args) {
int number =  -10 ;
    while (number <= 10) {
        if ( number %  2 == 0)
            System.out.println(number);
        number =  number + 1 ;
    }
}//main
}//Wrong
```

Uppgift 8

```
import javax.swing.*;
import java.util.*;
import java.text.*;
public class StatligSkatt {
    public static void main (String[] arg) {
        boolean done = false;
        while (!done) {
            String indata = JOptionPane.showInputDialog("Ange beskattningsbar förvärvsinkomst "
                                                        + " och kapitalinkomst :");
            if (indata == null)
                done = true;
            else {
                Scanner sc = new Scanner(indata);
                int inkomst = sc.nextInt();
                int kapital = sc.nextInt();
                if (inkomst < 0 || kapital < 0)
                    JOptionPane.showMessageDialog(null, "Ogiltig indata! Kan inte vara negativ!");
                else {
                    int skatt = computeTax(inkomst, kapital);
                    JOptionPane.showMessageDialog(null, "Inkomstskatt är :" + skatt + " kr");
                }
            }
        }
    }//main

    public static int computeTax(int incomeOfSalary, int incomeOfCapital) {
        double tax = 0;
        if ( incomeOfSalary > 438900 && incomeOfSalary <= 638500)
            tax = ( incomeOfSalary - 438900) * 0.20;
        else if ( incomeOfSalary > 638500)
            tax = ( 638500 - 438900) * 0.20 + ( incomeOfSalary - 638500) * 0.25;
        tax = tax +   incomeOfCapital * 0.3;
        return (int) tax;
    }//computeTax
}//StatligSkatt
```

Uppgift 9

```
import java.util.*;
public class DominoModel {
    private ArrayList<Tile> titles;
    private static Random random = new Random();
    public DominoModel() {
        titles = new ArrayList<Tile>();
        for (int i = 0; i <= 6; i++) {
            for (int j = i; j <= 6; j++) {
                titles.add(new Tile(i, j));
            }
        }
    }//constructor

    public int nrOfTitles() {
        return titles.size();
    }

    public Tile pickTile() {
        if (titles.size() == 0)
            return null;
        int i = random.nextInt(titles.size());
        return titles.remove(i);
    }//pickTitle

    public boolean match(Tile tile1, Tile tile2) {
        return (tile1.getLeft()  == tile2.getLeft()  ||
                tile1.getLeft()  == tile2.getRight() ||
                tile1.getRight() == tile2.getLeft()  ||
                tile1.getRight() == tile2.getRight());
    }//match
}//DominoModel
```

Uppgift 10

a)

```
public static boolean isValid(String password) {
    if (password.length() < 8)
        return false;
    int nrOfUpperCase = 0; int nrOfLowerCase = 0; int nrOfNonLetters = 0;
    for (int i = 0;  i < passw.length();  i = i + 1) {
        char ch = passw.charAt(i);
        if (Character.isUpperCase(ch) )
            nrOfUpperCase++;
        if (Character.isLowrrCase(ch) )
            nrOfLowerCase++;
        if (!Character.isLetterOrDigit(ch))
            nrOfNonLetters++;
    }
    if (nrOfUpperCase > 0 && nrOfLowerCase > 0 &&  nrOfNonLetters > 1)
        return true;
    else
        return false;
}  //isValid
```

b)

```
public static double[][] toMatrix(double[] scanned, int nrRows) {
    int nrCols = scanned.length / nrRows;
    double[][] view = new double[nrRows][nrCols];
    int pos = 0;
    for (int row = 0; row < nrRows; row = row + 1) {
        if (row % 2 == 0) {
            for (int col = 0; col < nrCols; col = col + 1) {
                view[row][col] = scanned[pos];
                pos++;
            }
        } else {
            for (int col = nrCols-1; col >= 0; col = col - 1) {
                view[row][col] = scanned[pos];
                pos++;
            }
        }
    }
    return view;
}//toMatrix
```

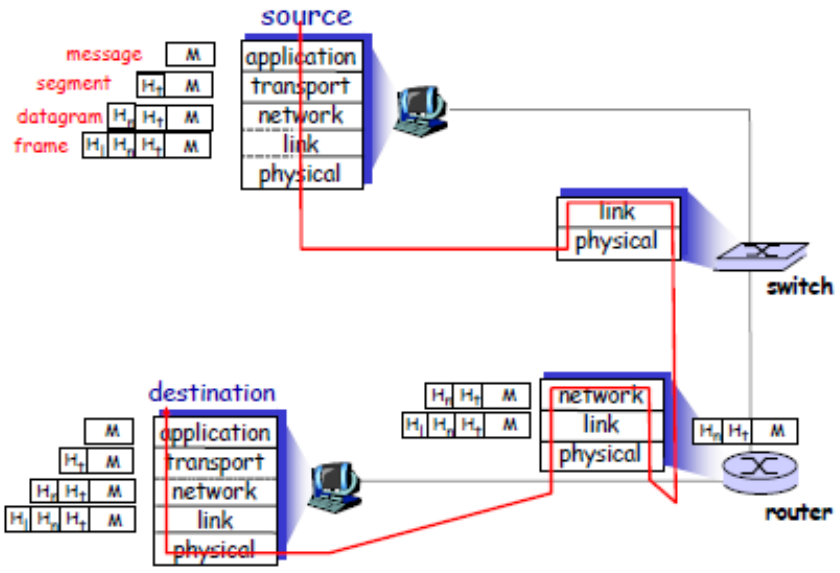
c)

```
public static double[][][] fromRGBtoCMYK(int[][][] picture) {
    double[][][] newPicture = new double[picture.length][picture[0].length][4];
    for (int row = 0; row < picture.length; row = row + 1) {
        for (int col = 0; col < picture[row].length; col = col + 1) {
            double maxOfRG = Math.max(picture[row][col][0]/255, picture[row][col][1]/255);
            double white = Math.max(maxOfRG, picture[row][col][2]/255);
            newPicture[row][col][0] = (white - picture[row][col][0]/255) / white;
            newPicture[row][col][1] = (white - picture[row][col][1]/255) / white;
            newPicture[row][col][2] = (white - picture[row][col][2]/255) / white;
            newPicture[row][col][3] = 1 - white;
        }
    }
    return newPicture;
}//fromRGBtoCMYK
```

Uppgift 11

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class TwinkleLable extends JLabel implements ActionListener {
    private boolean textVisible = false;
    private String text;
    private javax.swing.Timer t = new javax.swing.Timer(500, this);
    public TwinkleLable(String text) {
        setHorizontalAlignment(CENTER);  //för att få texten centrerad, default är LEFT
        setBackground(Color.YELLOW);
        setOpaque(true);
        this.text = text;
    } //constructor

    public void setTwinkleOn() {
        t.restart();
        setText(text);
        textVisible = true;
    }

    public void setTwinkleOff() {
        t.stop();
        setText("");
        textVisible = false;
    }

    public void actionPerformed(ActionEvent e) {
        if (textVisible) {
            setText("");
            textVisible = false;
        } else {
            setText(text);
            textVisible = true;
        }
        repaint();
    }//actionPerformed
}// TwinkleLable
```

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class ShowTwinkleLable extends JFrame implements ActionListener{
    private JButton start, stop;
    private TwinkleLable lable = new TwinkleLable("Varning!!!");

    public ShowTwinkleLable(){
        JPanel buttons = new JPanel();
        buttons.setLayout(new GridLayout(1, 2, 10, 10));
        start = new JButton("Start");
        start.addActionListener(this);
        buttons.add(start);
        stop = new JButton("Stop");
        stop.addActionListener(this);
        buttons.add(stop);
        setLayout(new GridLayout(2, 1, 10, 10));
        add(lable);
        add(buttons);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
    }//constructor

    public void actionPerformed(ActionEvent e){
        if (e.getSource() == start)
            lable.setTwinkleOn();
        if (e.getSource() == stop)
            lable.setTwinkleOff();
        lable.repaint();
    }//actionPerformed
}//ShowTwinkleLable

public class MainTwinkle {
    public static void main(String[] args){
        ShowTwinkleLable r = new ShowTwinkleLable();
        r.setSize(150, 150);
        r.setVisible(true);
    }
}//MainTwinkle
```


source
message
segment
datagram
frame
application
transport
network
link
physical
link
physical
switch
destination
application
transport
network
link
physical
network
link
physical
router