Tentamen för TDA143 PROGRAMMERADE SYSTEM

DAG: 16-03-15 TID: 8:30 – 13:30

Ansvarig:		Christer Carlsson, ankn 1038
Förfrågningar:		Christer Carlsson
Resultat:		erhålls via Ladok
Allmän info:		tentamen är uppdelad i två delar: del 1 omfattar översikt av datateknik och del 2 omfattar programmering
Betygsgränser:	3:a	10 poäng på del 1 och 26 poäng på del 2
	4:a	13 poäng på del 1 och 36 poäng på del 2
	5:a	16 poäng på del 1 och 48 poäng på del 2
	maxpoäng	20 poäng på del 1 och 60 poäng på del 2
Siffror inom parentes:		anger maximal poäng på uppgiften.
Granskning:		Måndag 11/4 kl 12-13 och tisdag 12/4 kl 12-13, rum 6128 i EDIT-huset.
Hjälpmedel:		En valfri lärobok i Java eller de på kursen utdelade föreläsningsanteckningarna (OH-bilderna) som behandlar **programmeringsdelen** av kursen.
		Förtydligande noteringar får finnas i boken resp. föreläsningsanteckningarna.
Var vänlig och:		Skriv tydligt och disponera papperet på lämpligt sätt.
		Börja varje uppgift på nytt blad. Skriv ej på baksidan av papperet.
Observera:		Uppgifterna är ej ordnade efter svårighetsgrad. Titta därför igenom hela tentamen innan du börjar skriva.
		Alla program skall vara väl strukturerade, lätta att överskåda samt enkla att förstå.
		Vid rättning av uppgifter där programkod ingår bedöms principiella fel allvarligare än smärre språkfel.
		På de programmeringsuppgifter som ingår i tentamenstesen kan vissa poäng erhållas även om ett fullständigt program inte redovisas, detta kräver dock att en algoritm som löser problemet presenteras.

LYCKA TILL!!!!

Uppgift 1.

Denna uppgift består av 10 påståenden. Frågorna skall besvaras med ”sant” eller ”falskt”. Ett korrekt svar ger 1 poäng, ett felaktigt svar ger -0.5 poäng och ett utelämnat svar ger 0 poäng. Totalt på uppgiften kan aldrig minuspoäng erhållas.

a) $8A_{16} = 138_{10} = 10011010_2$

b) Enligt *Moores lag* fördubblas antal transistorer som ryms på en given yta var 36:e månad.

c) När man ser på film över Internet, och måste överföra stora datamängder, är UDP *(*User Datagram Protocol) ett lämpligare transportprotokoll än TCP (Transmission Control Protocol).

d) Transportlagret (*transport layer*) och länklagret (*link layer*) kommunicerar direkt med varandra.

e) Nedanstående algoritm har tidskomplexiteten $\Theta(n^2)$.

```
int sum = 0;
for (int i =1; i <= n;  i = i + 1)
    sum = sum + i*i;
for (int i =1; i <= n;  i = i + 1)
    for (int j =1; j <= n;  j = j + 1)
        sum = sum + i*j;
```

f) En *glupsk algoritm* hittar alltid den optimala lösningen.

g) I en databas eftersträvar man *redundans* för att öka tillgängligheten till den information som finns lagrad i databasen.

h) Korta utvecklingscykler är en förutsättning för att mjukvaruföretag skall bli framgångsrika.

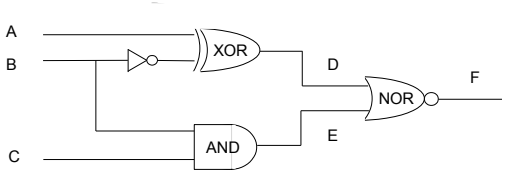
i) Enligt Volvo Cars, är 80-90% av alla inovationer inom bilindustrin relaterad till elektronik.

j) Fresnel-effekten, som är viktig att ta hänsyn till för att få realism i datorgenerade miljöer, innebär att reflektioner i vissa föremål/material endast uppstår när föremålet observeras vid en viss vinkel.

(10 poäng)

Uppgift 2.

Ange sanningstabellen för den logiska kretsen nedan:



(2 poäng)

Uppgift 3.

Antag att ett binärt träd har sex noder med namnen A - F. Det gäller att A och B är syskon, E och F är syskon, C är förälder till A, B är förälder till D samt A är ett löv. Rita ut hur trädet ser ut.

(2 poäng)

Uppgift 4.

I Java är heltalet 2147483647 det största värdet som kan lagras i en variabel av typen **int**. Exekveras satsen

```
int value = 2147483647 + 1;
```

kommer variabeln value att få värdet -2147483648 ! Förklara varför! (2 poäng)

Uppgift 5.

Att ha sin dator ansluten till Internet är inte helt riskfritt. Beskriv kort vilka faror man utsätts för och hur man kan skydda sig.

(2 poäng)

Uppgift 6.

Antag att du har följande databastabeller:

Meny:

Maträtt	Typ	Pris	Kalorier
Kokt ishavstorsk med hackat ägg	Fisk	139	460
Pannbiff med stekt lök	Kött	109	760
Omelett	Vegetarisk	89	250
Wienerschnitzel av gödkalv	Kött	119	550

Luncherbjudanden:

Maträtt	Typ	Pris	Kalorier
Stekt sill	Fisk	79	440
Wallenbergare med ärtor och potatispuré	Kött	89	760
Pasta bolognese	Kött	69	650

Hur ser tabellen ut som erhålls genom nedanstående SQL-query?

```
SELECT Maträtt, Pris
FROM Meny, Luncherbjudanden
WHERE Kalorier < 500
```

(2 poäng)

DEL 2: Programmeringsteknik

Uppgift 7.

a) Betrakta nedanstående program:

```
public class Uppgift7a {
    public static void main(String[] args) {
        System.out.println(mystery("B3A8"));
    }//main
    public static int  mystery(String value) {
        String str ="0123456789ABCDEF";
        int res = 0;
        for (int  i = 0; i < value.length(); i = i + 1) {
            res = 16*res + str.indexOf(value .charAt(i));
        }
        return  res;
    }//mystery
}//Uppgift7a
```

i) Vad blir utskriften när main-metoden exekveras?

ii) Beskriv i ord vilket syfte metoden mystery har (dvs vilken uppgift metoden utför). (2 poäng)

b) Betrakta följande klasser:

```
public class Utils {
    public static void changeValue(int[] arr, int i, int k) {
        int temp = arr[i];
        arr[i] = arr[k];
        arr[k] = temp;
    }//changeValue
    public static void changeValue(int a, int b) {
        int temp = a;
        a = b;
        b = temp;
    }//changeValue
}//Utils

public class Main {
    public static void main(String[] args) {
        int[] vect = {2, 4, 6, 8, 10};
        Utils.changeValue(vect, 0, 1);
        System.out.println(vect[0] + " " + vect[1]);
        Utils.changeValue(vect[2], vect[3]);
        System.out.println(vect[2] + " " + vect[3]);
    }//main
}//Main
```

Vad skrivs ut när main-metoden i klassen Main exekveras? (2 poäng)

c) Betrakta nedanstående metod:

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

Avsikten är att metoden skall returnera antalet siffror som ingår i heltalet number. Metoden går att exekvera, men ger inte alltid rätt resultat. När blir resultatet galet? Rätta till i koden! (2 poäng)

Uppgift 8.

Förtjänsten vid försäljning av en aktiepost kan beräknas enligt följande:

profit = ((NS x SP) - SC) – ((NS x PP) + PC)

där NS är antalet aktier, SP försäljningspris per aktie, SC betald provision vid försäljning, PP är inköpspriset per aktie och PC betald provision vid inköp.

Skriv ett program för att upprepade gånger läser in antalet aktier, inköpspris, provision vid inköp, försäljningspris samt provision vid försäljning och skriver ut resultatet av aktieförsäljningen. Du får själv välja om du vill göra in- och utmatning via dialogrutor eller använda System.in respektive System.out. Exekveringen av programmet avbryts vid användning av dialogrutor genom att användaren trycker på Cancel-knappen och vid användning av System.in genom att användaren lämpligen ger ctrl z. Ett körningsexempel med användning av dialogrutor visas nedan.

För att få full poäng på uppgiften:

- skall programmet innehålla en metod

  public static double profit(**int** NS, **double** PP, **double** PC, **double** SP, **double** SC)

  som beräknar och returnerar resultatet av försäljningen.
- skall all indata ges i en och samma inläsningssats (dvs ett Scanner-objekt skall användas).
- skall felutskriften OGILTIG INDATA! skrivas ut om användaren ger negativa indatavärden.
- skall resultatet av försäljningen presenteras med exakt 2 decimaler.

<u>Körningsexempel med användning av dialogrutor</u>:

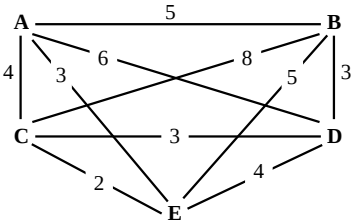
(10 poäng)

Uppgift 9

Ett klassiskt datalogisk problem, som omnämts under kursen, är det så kallade *handelsresandens problem*. Problemet kan formuleras enligt följande:

> En handelsresande skall besöka N givna städer, som är sammanbundna genom ett vägnät. Bestäm i vilken ordning handelsresanden skall besöka dessa städer så att den sammanlagda resvägen blir så kort som möjlig. Resan påbörjas och avslutas i samma stad och de övriga städerna skall besökas exakt en gång.

Längden av en resväg är summan av de N inviduella avstånden mellan två på resvägen närbelägna städer. I exemplet i figuren nedan antas handelsresanden starta i A. En möjlig resväg är då A-D-E-B-C-A, som har längden 6+4+5+8+4 = 27. En annan möjlig och betydligt bättre resväg är A-E-C-D-B-A, som har längden 3+2+3+3+5 = 16.



Att lösa handelsresandens problem är mycket svårare än vad man vid en första anblick kan tro. För att få en korrekt lösning måste längden av *samtliga möjliga resvägar* beräknas. Antalet möjliga resvägar växer emellertid exponentiellt med avseende på antalet städer. Denna tillväxt innebär att det redan vid ett ganska litet antal städer blir omöjligt att bestämma den korrekta lösningen. Om det för en dator t.ex. tar 0.001 sekunder för att bestämma samtliga resvägar för 10 städer, så tar det 1.0 sekunder för 20 städer, ca. 18 minuter för 30 städer, 12 dagar för 40 städer och 366 århundranden för 60 städer!

Handelsresandens problem är inget akademiskt problem utan är i högsta grad verkligt inom logistik. Har vi t.ex. en robot som borrar ett stort antal hål i ett arbetsstycke, vill vi naturligtvis att roboten skall ägna så mycket tid som möjligt åt själva borrningen och inte uppta tiden med att förflytta sig fram och tillbaka över arbetsstycket.

De problem som i likhet med handelsresandens problem har en exponentiell tillväxt går under namnet NP-komplexa problem. För denna klass av problem finns inga praktiskt användbara algoritmer för att finna den optimala lösningen. Istället får man söka finna användbara algoritmer som ger lösningar i närheten av den optimala. En sådan algoritm är följande:

> Från den stad handelsresanden befinner sig går resan till den stad som ligger närmast av de städer som ännu inte blivit besökta. Denna stad blir nu den nya stad där handelsresanden befinner sig. Sedan upprepas förfarandet tills alla städer blivit besökta, varefter handelsresanden återvänder till den stad där resan startade.

I denna uppgift skall du skriva ett program som ger en approximativ lösning till handelsresandens problem enligt den ovan beskrivna algoritmen.

a) Skriv en klass City som avbildar städer. Klassen skall ha följande attribut:

- namn
- geografisk position
- besökt

Klassen skall innehålla

public City(String name, Point position)	konstruktor som har namn och position som argument. När en stad skapas är den ännu inte besökt.
public void setVisited()	sätter att staden blivit besökt.
public boolean getVisited()	returnerar huruvida staden blivit besökt eller ej.
public String getName()	returnerar stadens namn.
public double distanceTo(City other)	returnerar avståndet mellan den aktuella staden och staden other.

Tips 1: Klassen Point, som representerar en punkt i det tvådimensionella planet, finns i paketet java.awt. De metoder du behöver ha kännedom om är:

double getX()	returnerar punktens x-koordinat
double getY()	returnerar punktens y-koordinat

Tips 2: Använd Pythagoras sats för att beräkna avståndet mellan två städer. (8 poäng)

b) Skriv en metod

public static City getClosesCity(ArrayList<City> cities, City presentCity)

som bland de *ännu ej besökta* objekten i listan cities returnerar det objekt som ligger närmast (har det kortaste avståndet till) objektet presentCity. Du får anta att presentCity har blivit besökt. Metoden skall placeras i en klass med namnet Salesman.

(6 poäng)

c) Utöka klassen Salesman med en metod

public static void calculateRoute(ArrayList<City> citiesToVisit, City startCity)

som tar en lista citiesToVist vilken innehåller de städer som handelsresanden skall besöka, samt en stad startCity vilken är staden där handelsresanden påbörjar sin resa. Metoden skall, med användning av ovan beskrivna algoritm, skriva ut handelsresandens resvägen samt resvägens längd. Givetvis skall du i din lösning nyttja metoden getClosesCity från föregående deluppgift.

Exempel:

När nedanstående main-metod exekveras

```
public static void  main(String[] arg) {
    ArrayList<City> cities = new ArrayList<City>();
    City start = new City("A", new Point(0, 0));
    cities.add(start);
    cities.add(new City("B", new Point(1, 8)));
    cities.add(new City("C", new Point(5, 2)));
    cities.add(new City("D", new Point(3, 7)));
    cities.add(new City("E", new Point(9, 3)));
    Salesman.calculateRoute(cities, start);
}//main
```

skall följande utskrift erhållas

```
Resvägen är: A - C - E - D - B - A
Totala ressträckan är: 27.01769870947848
```

(8 poäng)

Uppgift 10

Det geometriska medelvärdet av n positiva tal $x_1,\ldots, x_n$ erhålls med hjälp av formeln

$$G=\sqrt[n]{x_1\cdot x_2\cdot\ldots\cdot x_n}$$

Exempel:

Det geometriska medelvärdet av 2 och 8 är

$$\sqrt{2\cdot 8}=4.0$$

Det geometriska medelvärdet av 1, 2 och 3 är

$$\sqrt[3]{1\cdot 2\cdot 3}=1.8171205928321297$$

Skriv en statisk metod

public static double geometricMean(**int**[] values)

som beräknar det geometriska medelvärdet för talen i heltalsvektorn values. Om values är **null** skall metoden kasta ett objekt av typen NullPointerException. Om values inte innehåller några element skall metoden kasta ett objekt av typen IllegalArgumentException. Om values innehåller något icke-positivt tal skall metoden kasta ett objekt av typen NumberFormatException.

Tips: $\sqrt[n]{k}=k^{\frac{1}{n}}$ (7 poäng)

Uppgift 11

Steganografi handlar om att skriva dolda budskap på ett sådant sätt att ingen - bortsett från avsändaren och den avsedda mottagaren - misstänker att det finns ett meddelande. I denna uppgift skall du skriva en metod för att avkoda en gråskalebild som finns dold i en färgbild.

En digital färgbild lagras, som bekant, på RGB-format. Varje bildpunkt utgörs av *tre* heltalsvärden i intervallet [0,255], där de enskilda värdena representerar intensiteten av färgerna rött, grönt respektive blått. En färgbild kan således avbildas med ett tredimensionellt fält av typen **int**[][][], där den första dimensionen definierar bildens höjd, den andra dimensionen bildens bredd och den tredje dimensionen de tre färgerna.

I en gråskalebild utgörs varje bildpunkt av *ett* heltalsvärde i intervallet [0, 255], där värdet representerar bildpunktens ljushet. En digital gråskalebild kan således representeras av ett tvådimensionellt fält av typen **int**[][].

Den gråskalebild som som du skall avkoda har gömts i den digital färgbild enligt följand tillvägagångssätt:

> Entalssiffran i bildpunkten för gråskalebilden döljas i den minst signifikanta siffran i komponenten som representerar blått i färgbilden, tiotalssiffran i bildpunkten för gråskalebilden döljas i den minst signifikanta siffran i komponenten som representerar grönt i färgbilden och hundratalssiffran i bildpunkten för gråskalebilden döljas i den minst signifikanta siffran i komponenten som representerar rött i färgbilden.

Exempel:

> En bildpunkt i gråskalebilden som har värdet 197 göms i en färgbildpunkt vars RGB-värden är [155, 213, 163] således enligt:

Ursprunglig färgbildpunkt				**Färgbildpunkt med gömd gråskalebildpunkt**		
Red	Green	Blue	göm värde 197	Red	Green	Blue
155	213	163	→	15**1**	21**9**	16**7**

> Modifikationen i färgbild är så liten att den inte kan upptäckas med blotta ögat.

Din uppgift är att skriva en metod

> **public static int**[][] decodeGrayPicture(**int**[][][] colourPicture)

som avkodar gråskalebilden som finns dold, enligt tillvägagångssättet ovan, i färgbilden colourPicture. Gråskalebilden som är dold har samma storlek som frägbilden.

Kommentar: För att kunna dölja meddelanden i digitala bilder måste man använda ett bildformat som komprimerar bilderna på ett sådant sätt att ingen information går förlorad vid komprimeringen. Formatet .png fungerar, men däremot inte .jpg och .gif.

(6 poäng)

Uppgift 12

Skriv ett program som skapar ett fönster, enligt figurerna nedan. I fönstret visas en cirkel som successivt ökar i storlek. Till att börja med är cirkelns radie 1 pixel. Cirkel är centrerad i fönstrets rityta. Cirkelns radie ökar med 1 pixel varje sekund. Denna ökning pågår tills cirkeln i sin helhet inte längre ryms på ritytan.

a) Skriv en klass GrowingCircle, som beskriver själva cirkeln. Denna klass skall vara en subklass till standardklassen JPanel. För att handha tillväxten av cirkelns radie har klassen en instans av klassen javax.swing.Timer.

b) Skriv en klass ShowGrowingCircle, som definierar ett fönster innehållande en instans av klassen GrowingCircle. Givetvis behövs också ett main-metod implementeras (antingen i klassen ShowGrowingCircle eller i en separat klass).

(9 poäng)