

**NTNU**  
**Norges teknisk-naturvitenskapelige**  
**universitet**

**Fakultetet for informasjonsteknologi,**  
**matematikk og elektroteknikk**

**Institutt for datateknikk og**  
**informasjonsvitenskap**

**BOKMÅL**



**LØSNINGSFORSLAG**  
**Kontinuasjoneksamen i TDT4110**  
**Informasjonsteknologi, grunnkurs**  
**Onsdag 10. august 2011**  
**9.00 – 13.00**

**Hjelpemidler: C**

Tilleggshefte I, "Introduksjon til HTML, CSS, JSP og MYSQL"  
Bestemt, enkel kalkulator: HP 30S eller Citizen SR270-X

### **Oppgave 1: Flervalgsoppgave (25 %)**

Bruk de to vedlagte svarskjemaene for å svare på denne oppgaven (ta vare på det ene selv). Du kan få nytt ark av eksamensvaktene dersom du trenger dette. Kun ett svar er helt riktig. For hvert spørsmål gir korrekt avkryssing 1 poeng. Feil avkryssing eller mer enn ett kryss gir  $-1/2$  poeng. Blankt svar gir 0 poeng. Du får ikke mindre enn 0 poeng totalt på denne oppgaven. Der det er spesielle uttrykk står den engelske oversettelsen i parentes.

1. Hva er en *tjener* i en klient-/tjenerarkitektur?

- a) **En tjener utfører tjenester på vegne av en klient.**
- b) En tjener etterspør tjenester fra klienter.
- c) En tjener er et program som vil ha noe utført på en annen maskin.
- d) Ingen av alternativene i a, b og c.

2. En del av et program består av en FOR-løkke, inne i en FOR-løkke, inne i en FOR-løkke. Alle løkkene løper gjennom tallene fra 1 til  $N$ . Tidsforbruket for denne programdelen vil være proporsjonal med:

- a)  $N^3$
- b)  $3N$
- c)  $3N \log N$

3. Hva er hovedoppgaven til en *navnetjener*?

- a) **Å oversette domenenavn til IP-adresser.**
- b) Å oversette IP-adresser til domenenavn.
- c) Å holde rede på definerte variabelnavn når et program kjører.
- d) Ingen av alternativene i a, b og c.

4. Kari har utviklet et programsystem i den bedriften der hun har vært ansatt. Kan hun ta med seg og videreutvikle denne programvaren hvis hun slutter i bedriften?

- a) Ja, i følge Åndsverksloven er det programmereren som har rettighetene til programvaren.
- b) **Nei, det er bedriften som eier programvaren.**
- c) Det er uavklart siden dette er ikke regulert av lovgiving. Rettighetene til programvaren må avtales mellom partene i hvert enkelt tilfelle.

5. Svart-boks testing er:

- a) Testing der man prøver ut systemets eller modulens indre oppbygging.
- b) **Testing der man ser bort fra systemets eller modulens indre oppbygning og kun tar i betraktning de ytre egenskapene.**
- c) Testing der man ser bort fra systemenes eller modulenes indre oppbygning og prøver ut hvordan de ulike delene i et system fungerer sammen.

6. Hva er riktig rekkefølge når ulike minnetyper sorteres etter synkende hastighet?

- a) **Registre, cache, primærminne, sekundærminne.**
- b) Cache, registre, primærminne, sekundærminne.
- c) Primærminne, cache, sekundærminne, registre.

7. Når kan man *ikke* benytte seg av binærsøkingsalgoritmen?

- a) Når datamengden er så stor at det vil ta for lang tid.
- b) Når det er stor sannsynlighet for at det man søker etter ikke finnes i datamengden.
- c) Når datamengden ikke er sortert.**
- d) Man kan alltid bruke binærsøkingsalgoritmen.

8. Hva er i lovverket ikke definer som *sensitive* personopplysninger:

- a) Etnisk bakgrunn.
- b) Politisk oppfatning.
- c) Filosofisk oppfatning.
- d) Alle alternativene i a, b og c er sensitive personopplysninger.**

9. Vi skal kode tegnene a-å, A-Å, 0-9, mellomrom, spørsmålstegn, utropstegn, punktum, kolon, semikolon og bindestrek. Hvor mange binære siffer (engelsk: bits) er nødvendig i kodene når alle tegn skal representeres med like mange binære siffer?

- a) 6
- b) 7**
- c) 8

10. Hva blir resultatet når 81 kodes binært (i 2-tallsystemet)?

- a) 1011011
- b) 1010101
- c) 1010001**
- d) Ingen av alternativene i a, b og c er riktige.

11. Hva tilsvarer ABCD i det heksadesimale tallsystemet (16-tallsystemet) i 10-tallsystemet?

- a) 42561
- b) 43981**
- c) 44981
- d) Ingen av alternativene i a, b og c er riktige.

12. QoS er forkortelse for:

- a) Quantity of Storage.
- b) Quality of Servers.
- c) Quality of Service.**
- d) Ingen av alternativene i a, b og c er riktige.

13. Vi har en sortert liste med 100 000 000 elementer. Ved binærsøking i denne listen, hvor mange sammenligninger må vi i verste fall gjøre?

- a) Omtrent 20.
- b) Omtrent 27.**
- c) Omtrent 34.

14. `<p>...</p>` i en HTML-fil definerer:

- a) **Et avsnitt.**
- b) Et adressefelt.
- c) En hyperlenke.
- d) Ingen av alternativene i a, b og c.

15.  $A \parallel \neg(B \ \&\& \ C)$  er usant (engelsk: false) når:

- a) **A false, B true, C true**
- b) A true, B true, C false
- c) A false, B false, C true
- d) Uttrykket er ikke usant for noen av alternativene i a, b og c.

16. Hva er *entropi*?

- a) **Et mål på informasjonsinnholdet i en melding.**
- b) En algoritme for effektiv koding av symboler.
- c) En feiltilstand som gjør at lesehodet på en harddisk ikke klarer å følge sporene nøyaktig og derfor blander data fra flere spor.
- d) Ingen av alternativene i a, b og c er riktige.

17. En URL (Uniform Resource Locator) har følgende format:

- a) **Protokoll, maskin, sti, fil**
- b) Protokoll, maskin, sti
- c) Domene, sti, fil
- d) Ingen av alternativene i a, b og c er riktige.

18. Anta at karakterene har følgende fordeling: A (12 %), B (24 %), C (35 %), D (20 %), E (7 %) og F (2 %). Hva er en korrekt Huffmankoding for A-F?

- a) **A: 001    B: 10    C: 11    D: 01    E: 0001    F: 0000**
- b) A: 010    B: 01    C: 10    D: 111    E: 1110    F: 1111
- c) A: 1111    B: 11    C: 1    D: 111    E: 11111    F: 111111
- d) Ingen av alternativene i a, b og c.

19. En IP-adresse er:

- a) **En numerisk adresse som identifiserer en datamaskin eller annen type enhet i et IP-nettverk.**
- b) En mekanisme som gjør det mulig å kontakte en Internett Provider, for eksempel når det oppstår feil i nettet.
- c) Et felt som identifiserer data i en relasjonsdatabase.

20. Hva er *inspeksjon* av programvare?

- a) Kjøring av programmer for å se om programvaren oppfyller kravene til systemet.
- b) Overvåking av variablenes verdier under programkjøring for å finne årsakene til feil.
- c) **Gjennomgang av programmer uten at de blir kjørt, med tanke på å finne feil og å forvise seg om at de er riktige.**

## Oppgave 2: Programforståelse (10 %)

Gitt følgende programkode som er lagret i filen conv.jsp:

```
<%!  
int conv(int n, int m) {  
    int[] temp = new int[10];  
    int i = 0;  
    while (n>0) {  
        temp[i] = n % m; // % gives the remainder from division  
        n = n/m;  
        i = i +1;  
    }  
    int res = f(temp, i-1);  
    return res;  
}  
  
int f(int[] list, int n) {  
    int res = 0;  
    for (int i =0;i<n; i++) {  
        res = res+ list[n-i+1];  
    }  
    return res;  
}
```

Hva blir resultatet (i variabelen *result*) når vi utfører koden:

```
int result = conv(100,3);
```

Løsning: result = 3
------------------------

### Oppgave 3: Programmering (15 %)

Annuitetslån er lån der man betaler et like stort beløp i det antall terminer som lånet løper. Ved slike lån går mye av dette terminbeløpet til renter i starten, etter hvert blir en stadig større andel av terminbeløpet avdrag (tilbakebetaling).

I tabellen under har vi vist hvordan et annuitetslån på 1000 kroner nedbetales over 3 terminer med en terminrente på 5 %.

Termin	Restlån	Terminbeløp	Rente	Avdrag
0	1000.0			
1	682.8	367.2	50.0	317.2
2	349.7	367.2	34.1	333.1
3	0	367.2	17.5	349.7

Det som betales i rente i en termin regnes ut med formelen:

$$\text{restlån} * i$$

der restlån er gjestående lån ved inngangen av terminen og

$i = \text{terminrente\_i\_prosent}/100$ .

Det faste terminbeløpet ( $a$ ) regnes ut med formelen:

$$a = \frac{P(1+i)(1 - (\frac{1}{1+i}))}{1 - (\frac{1}{(1+i)^n})}$$

der  $P$  er lånebeløpet,  $i$  er terminrenten ( $\text{terminrente\_i\_prosent}/100$ ) og  $n$  er antall terminer.

Bruk `Math.pow(x,y)` i JSP for å beregne  $x^y$ .

- a) (5 %) Skriv en metode `rente` som tar inn parameterne `laanBelop` og `renteFot` (terminrente i prosent), og returnerer renten som påløper i terminen. Alle parametere skal være av typen desimaltall.

```
double rente(double belop, double renteFot) {
    return belop * renteFot/100;
}
```

- b) (5 %) Skriv en metode `termBelop` som tar inn parameterne `laanBelop`, `renteFot` og `antallTerminer`, og som returnerer beløpet som skal betales etter hver termin.

```
double termBelop(double laanBelop, double renteFot, int antallTerminer) {
    double i = renteFot/100;
    double over = laanBelop*(1+i)*(1-(1/(1+i)));
    double under = 1-(1/Math.pow(1+i,antallTerminer));
    return over/under;
}
```

- c) (10 %) Skriv en metode *restLaan* som tar inn parameterne *laanBelop*, *renteFot*, *antallTerminer* og *termin*, og som returnerer restlånet etter at det har gått *termin* terminer. Dersom lånet er nedbetalt etter den aktuelle terminen, skal det returneres et restlån lik 0.

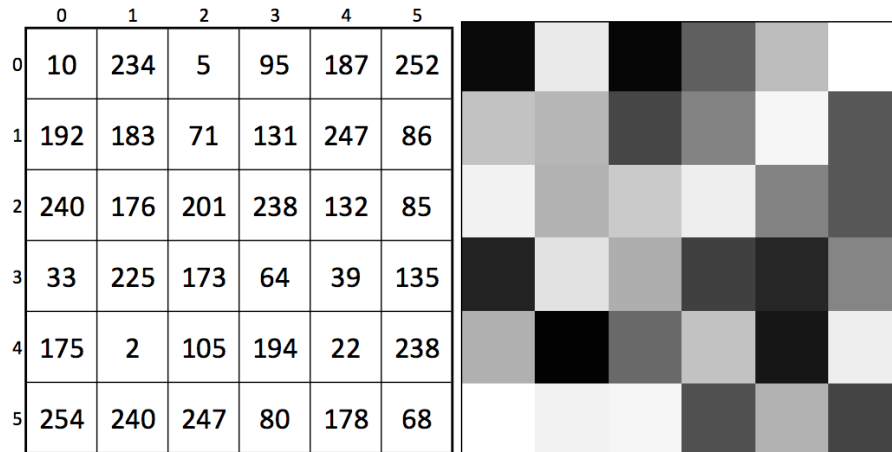
Dersom vi kaller opp `restLaan(1000, 5, 3, 2)` skal funksjonen returnere 349,7 som er restlånet etter to terminer i eksemplet som er vist i tabellen over.

Det er hensiktsmessig å gjenbruke funksjonene du laget i oppgave a og b. Du kan *bruke* funksjoner fra andre deloppgaver selv om du ikke har klart å løse deloppgaven der du skal lage funksjonen.

```
double restLaan(double laanBelop, double renteFot, int antTerminer, int
termin) {
    double tb = termBelop(laanBelop, renteFot, antTerminer);
    double saldo = laanBelop;
    for (int i=1;i<=termin;i++) {
        double termRente = rente(saldo, renteFot);
        saldo = saldo - (tb - termRente);
    }
    if (saldo < 0 ) {
        saldo = 0;
    }
    return saldo;
}
```

#### Oppgave 4: Programmering (50 %)

I denne oppgaven skal du programmere metoder brukt til analyse og behandling av kvadratiske bilder. Et bilde blir lagret i en kvadratisk tabell av heltall, der hvert punkt i bilde kan ha verdi fra 0 til 255 (8-bits) som angir gråtonen i bildet. Verdien 0 angir fargen svart, verdien 255 angir fargen hvit, mens alle verdiene mellom angir gråtoner som blir lysere med høyere verdi. Figur 1 viser hvordan et bilde på størrelse 6x6 er representert som en tabell (til venstre i figuren) og hvordan et bilde med gråtoner generert fra tallene i tabellen ser ut (til høyre i figuren).



Figur 1: Gråtonebilde lagret som kvadratisk tabell (venstre) og vist som bilde (høyre)

- a) (5%) Skriv metoden *random\_picture* som tar inn et heltall *size*, og returnerer en kvadratisk tabell av heltall på størrelse *size* som inneholder tilfeldige gråtoner (tall fra 0 til 255). Tabellen som er vist i figur 1 har *size* 6 etter som tabellen har størrelsen 6x6.

```
int[][] random_picture (int size) {
    int[][] picture = new int[size][size];
    for (int i=0; i<picture.length;i++) {
        for (int j=0; j<picture.length;j++) {
            picture[i][j] = (int) (Math.random()*256);
        }
    }
    return picture;
}
```

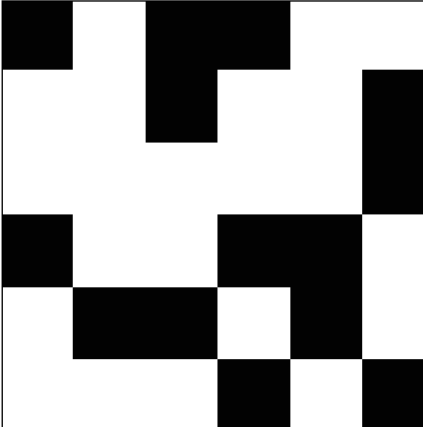
- b) (5%) Skriv metoden *test\_picture* som tar inn en todimensjonal tabell av heltall, *table*. Funksjonen skal returnere sannhetsverdien *true* hvis tabellen er kvadratisk samt at tabellen ikke inneholder noen ulovlige tallverdier.

```
boolean test_picture (int[][] table) {
    if (table.length!=table[0].length) { //test if table is square
        return false;
    } else {
        for (int i=0; i<table.length;i++) {
            for (int j=0; j<table.length;j++) {
                if (table[i][j]<0 || table[i][j]>255) {
                    return false;
                }
            }
        }
    }
    return true;
}
```



- c) (10%) Skriv metoden *filter\_black\_and\_white* som tar inn en todimensjonal tabell av heltall, *table*, samt et tall, *threshold*. Metoden skal returnere en endret tabell som er en svart/hvitt versjon av bildet representert i *table*. Alle verdiene i tabellen som er lavere enn *threshold* skal få verdien 0 (svart), mens alle andre verdier skal få verdien 255 (hvit). Kalles metoden *filter\_black\_and\_white* på tabellen fra figur 1 med *threshold* satt til 127, gir det resultatet som vist i figur 2.

	0	1	2	3	4	5
0	0	255	0	0	255	255
1	255	255	0	255	255	0
2	255	255	255	255	255	0
3	0	255	255	0	0	255
4	255	0	0	255	0	255
5	255	255	255	0	255	0



Figur 2. Resultatet etter å ha kjørt *filter\_black\_and\_white* på tabell fra figur 1

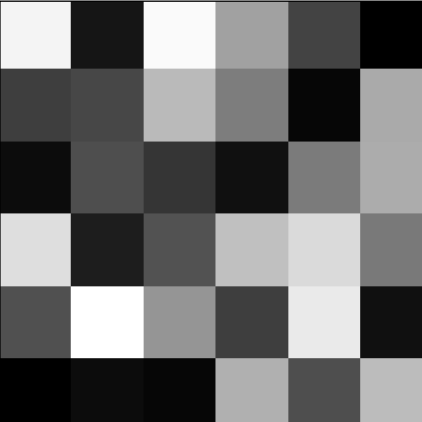
```

void filter_black_and_white(int[][] table, int threshold) {
    for (int i=0; i<table.length;i++) {
        for (int j=0; j<table.length;j++) {
            if (table[i][j]<threshold) {
                table[i][j]=0;
            } else {
                table[i][j]=255;
            }
        }
    }
}

```

- d) (10%) Skriv metoden *filter\_inverse*, som tar inn en todimensjonal tabell av heltall, *table*, og returnerer en tabell der innholdet i tabellen er invertert, dvs. alt som er lyst på bilde skal bli mørkt og motsatt. Figur 3 viser resultatet etter å ha kjørt *filter\_inverse* på tabellen fra figur 1.

	0	1	2	3	4	5
0	245	21	250	160	68	3
1	63	72	184	124	8	169
2	15	79	54	17	123	170
3	222	30	82	191	216	120
4	80	253	150	61	233	17
5	1	15	8	175	77	187



Figur 3. Resultat etter å ha kjørt *filter\_inverse* på tabell fra figur 1.

```

void filter_inverse(int[][] table) {
    for (int i=0; i<table.length;i++) {
        for (int j=0; j<table.length;j++) {
            table[i][j]=255-table[i][j];
        }
    }
}

```

- e) (10%) Skriv metoden *histogram* som tar inn en todimensjonal tabell av heltall, *table*, og returnerer en endimensjonal tabell som inneholder antall punkter i bildet (*table*) som har en bestemt gråtone. For bildet i figur 3 vil histogrammet for eksempel vise at bildet har 1 punkt med verdi 1, 1 punkt med verdi 3, 2 punkter med verdi 8, 2 punkter med verdi 15, 2 punkter med verdi 17 osv.

```

int[] histogram(int[][] table) {
    int[] histogram_table = new int[256];
    for (int i = 0; i<table.length;i++) {
        for (int j = 0; j<table.length;j++) {
            histogram_table[table[i][j]]++;
        }
    }
    return histogram_table;
}

```

- f) (10%) Bruk metodene ovenfor (selv om du ikke har løst deloppgavene) til å lage et JSP-script som gjør følgende:
- i. Lag et bilde med tilfeldig innhold med størrelse 10x10 punkter.
  - ii. Test om bildet er kvadratisk og at det ikke inneholder ulovlige verdier. Hvis bildet ikke blir godkjent, avslutt skriptet.
  - iii. Inverter bildet.
  - iv. Lag et histogram av bildet.
  - v. Skriv ut alle innslagene i histogrammet som har verdien 3 eller mer på følgende måte: Gråtone [gråtoneverdi] : [antall punkter] <linjeskift> .
  - vi. Gjør om bildet til sort/hvitt med terskelverdi 100.
  - vii. Skriv ut innholdet (verdiene) i bildet ved hjelp av en HTML-tabell der border skal være 1 og cellpadding skal være 10.

Eksempel på utskrift fra JSP-scriptet :

Gråtone 109: 3  
 Gråtone 135: 3  
 Gråtone 170: 3

255	0	0	0	255	255	255	0	255	255
0	0	255	0	255	255	255	255	255	0
255	255	0	255	0	0	255	255	0	0
0	0	255	255	255	255	0	255	255	0
255	0	0	0	255	0	0	255	255	255
0	255	255	255	0	255	0	0	255	0
255	0	0	255	255	0	0	0	0	255
255	255	255	255	0	255	255	0	255	255
255	0	255	255	255	255	255	255	255	255
0	255	255	0	255	255	255	255	0	255

```

<%
int[][] table = random_picture(10);

if (test_picture(table)) {
    filter_inverse(table);
    int[] histogram = histogram(table);
    for (int i=0;i<histogram.length;i++) {
        if (histogram[i]>2) {
            out.println("Gråtone "+i+": "+histogram[i]+"<br>");
        }
    }
    filter_black_and_white(table,100);
    out.println("<br><table border=1 cellpadding=10>");
    for (int i=0;i<table.length;i++) {
        out.println("<tr>");
        for (int j=0;j<table.length;j++) {
            out.println("<td>"+table[i][j]+"</td>");
        }
        out.println("<tr/>");
    }
    out.println("</table><br><br>");
}
%>

```