NTNU
Norges teknisk-naturvitenskapelige universitet

Fakultetet for informasjonsteknologi, matematikk og elektroteknikk

Institutt for datateknikk og informasjonsvitenskap

BOKMÅL

Sensurfrist: 21. januar 2011

Avsluttende eksamen i TDT4105 Informasjonsteknologi, grunnkurs

Tirsdag 21. desember 2010
9.00 – 13.00

Faglig kontakt under eksamen:
Jørn Amundsen (918 97 897), Roger Midtstraum (995 72 420)

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

Sensur:
Resultater gjøres kjent på http://studweb.ntnu.no.

Oppgavesettet inneholder 4 oppgaver. Det er angitt i prosent hvor mye hver oppgave og hver deloppgave teller ved sensur. Les igjennom hele oppgavesettet før du begynner å lage løsning. Disponer tiden godt! Gjør rimelige antagelser der du mener oppgaveteksten er ufullstendig, skriv kort hva du antar.

En liste over vanlige innebygde funksjoner i Matlab er vedlagt.

Svar kort og klart, og skriv tydelig. Er svaret uklart eller lenger enn nødvendig trekker dette ned.

Lykke til!

Innhold:

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 betyr det at et dataprogram er opphavsrettslig beskyttet?
   a) Det kan ikke kopieres og brukes uten at rettighetshaver får betalt.
   b) Det kan ikke kopieres og brukes uten at rettighetshaver blir informert.
   c) Det kan ikke kopieres og brukes uten rettighetshavers samtykke.

2) Hva defineres som en sensitiv personopplysning?
   a) Informasjon om kjøp i en nettbutikk.
   b) Medlemskap i et idrettslag.
   c) Både a og b.
   d) Verken a eller b.

3) Kari bryter seg inn på brukerkontoen til Ola, kopierer musikk som Ola har laget, og selger denne musikken på internett. Hvilke lover har Kari brutt?
   a) Lov om opphavsrett til åndsverk og Straffeloven.
   b) Straffeloven.
   c) Lov om opphavsrett til åndsverk.
   d) Ingen lover.

4) 01010101 i binærtallsystemet, er det samme som (i titallsystemet)?
   a) 105
   b) 95
   c) 85

5) 4095 (i titallsystemet) skal kodes heksadesimalt (16-tallsystemet). Hvor mange siffer blir det i resultatet?
   a) 5
   b) 4
   c) 3

6) En ER-modell:
   a) Viser hvilke entiteter og relasjoner som faktisk er lagret i en database.
   b) Beskriver informasjonsstrukturen i en database.
   c) Beskriver strukturen i et ERP-system (Enterprise Resource Planning System).

7) Anta at karakterene har følgende fordeling: A (7 %), B (20 %), C (35 %), D (25 %), E (8 %) og F (5 %). Hva er riktig Huffmannkoding for A-F?

a)	**11110 (A)**	**110 (B)**	**0 (C)**	**10 (D)**	**1110 (E)**	**11111 (F)**
b)	001 (A)	010 (B)	011 (C)	100 (D)	101 (E)	111 (F)
c)	110 (A)	01 (B)	0 (C)	1 (D)	11 (E)	111 (F)

8) I Matlab, hva blir resultatet av: 1+2/2–1?
   a) 0,5
   b) 1
   c) 3

9) En melding består av resultatet av 100 myntkast (mynt/krone). En annen melding består av resultatet av 100 terningkast (1–6 øyne). Hvilken melding har høyest entropi?
   a) Mynt-meldingen.
   b) Terning-meldingen.
   c) De har samme entropi.

10) Hvilket av alternativene er gyldig CSS?
   a) style="font-weight=bold"
   b) style="font-weight: bold, color: red"
   c) style="font-weight: bold;"

11) Hva er den *viktigste* oppgaven til en nettverkssvitsj?
   a) Sørge for at alle påkoblede maskiner mottar alle datapakker.
   b) Overvåke og stenge ute uønsket nettverkstrafikk.
   c) Fordele datatrafikken slik at datapakkene kommer frem til riktig mottaker.

12) Vi har en sortert liste med 5 000 000 elementer. Ved binærsøking i denne listen, hvor mange sammenligninger må vi i verste fall gjøre?
   a) Omtrent 20
   b) Omtrent 30
   c) Omtrent 40

13) For et større nettsted, hva er den *viktigste* fordelen ved å samle CSS-stiler i en stilfil?
   a) Det tvinger HTML-filene til å bruke de samme stilene og gir et konsistent utseende.
   b) Det tar mindre plass enn å gjenta de samme stilene i mange HTML-filer.
   c) Det gjør det enkelt å ha felles stiler for mange HTML-filer.

14) `<a…>…</a>` i en HTML-fil definerer:
   a) Et adressefelt.
   b) Et avsnitt.
   c) En hyperlenke.

15) Hvilket av disse er et *ikke-funksjonelt* krav?
   a) Systemet skal registrere nye brukere.
   b) Systemet skal kryptere persondata.
   c) Systemet skal gi en oversikt over ledige seter.

16) Testing av programkode vil si at man:
   a) Går gjennom programkoden og prøver å finne feil.
   b) Lar noen utvalgte brukere prøve systemet.
   c) Kjører programkoden og sjekker at forholdet mellom inndata og utdata er som forventet.

17) Hva er fordelen med *asymmetrisk* kryptering fremfor symmetrisk kryptering?
   a) Man kan publisere den ene nøkkelen.
   b) Det er vanskeligere å knekke koden fordi man må finne to ulike nøkler.
   c) Man kan være sikker på identiteten til den som har sendt meldingen (autentisering).

18) Hva er nettverkstjenester?
   a) E-post, filoverføring og trådløst nettverk.
   b) Fildeling, WWW og e-post.
   c) Filoverføring, TCP/IP og WWW.

19) CPU er forkortelse for:
   a) Central Pipelining Unit.
   b) Coordinating Processor Unit.
   c) Central Processing Unit.

20) Et tall av typen double lagres i 8 byte. Omtrent hvor mange slike tall kan lagres i en gigabyte (GB)?
   a) Ca. 134 000.
   b) Over 100 millioner.
   c) Omtrent en milliard.

21) Hvilken betydning har *klasse* (eng: class) i forbindelse med HTML/CSS:
   a) Det definerer klasser av lignende tags, for eksempel klassen av overskrift-tags (<h1>, <h2>, etc.).
   b) Det lar oss skille mellom viktige tags (head, body, etc) og ubetydelige tags (, etc.).
   c) Det lar oss definere en delmengde av alle forekomster av en tag, som vi kan gi spesiell behandling.

22) På hvilken måte skiller harddisk seg fra primærminne (RAM):
   a) Det tar lengre tid å skrive eller lese.
   b) Dataene er sikre mot strømbrudd.
   c) Verken a eller b.
   d) Både a og b.

23) `A && (B || C)` er true (sant) for:
   a) A true, B true, C false
   b) A true, B false, C false
   c) A false, B true, C true

24) Hovedgrunnen til at vi deler opp lengre programmer ved å bruke funksjoner:
   a) Det går raskere å kjøre programmet.
   b) Programmet får høyere funksjonalitet.
   c) Det er lettere å forstå programmet.

25) Hva stemmer *ikke* om kommentarer i programkode?
   a) Programmet tar større plass på harddisken.
   b) De gjør det lettere å forstå koden.
   c) Programmet kjører langsommere.

Oppgave 2: Programforståelse (10 %)

Følgende funksjoner er definert:

```
function y = a(x)
  x = x + 1;
  y = 1 + x * 2;
end
```

```
function y = b(n)
  if n < 40
    y = 2 * n;
  elseif n < 10
    y = n;
  else
    y = 1;
  end
end
```

```
function z = c(w)
  z = b(a(w));
  if z < 10
    z = z + w;
  end
end
```

a) (2 %) Hva er verdien til *x* etter at vi har kjørt `x = 2; x = a(x)`? **7**

b) (2 %) Hva er verdien til *x* etter at vi har kjørt `x = 2; x = b(x)`? **4**

c) (2 %) Hva er verdien til *x* etter at vi har kjørt `x = 2; x = c(x)`? **14**

d) (2 %) Hva er verdien til *x* etter at vi har kjørt `x = 20; x = b(x)`? **40**

e) (2 %) Hva er verdien til *x* etter at vi har kjørt `x = 50; x = b(x) + c(x)`? **52**

Oppgave 3: Programmering (15 %)

I skihopp gis det poeng for hopplengde og stil.

a) (5 %) Poeng for hopplengde beregnes med følgende formel:

distance_points = 60 + (*jump_distance* – *kpoint*) ⋆ *meter_value*

Tallene *kpoint* og *meter_value* er fastsatt for hver hoppbakke.

Hoppbakken i Granåsen har *kpoint* 124 og *meter_value* 1,8. En skihopper som hopper 140 meter i denne bakken vil få 60 + (140 – 124) ⋆ 1,8 = 88,8 lengdepoeng (distance points).

Skriv en funksjon *distance_points* som tar inn parametrene *distance* (hopplengde), *kpoint* (k-punkt) og *meter_value* (meterverdi), og returnerer lengdepoeng.

```
function points = distance_points(jump_distance, kpoint, meter_value)
    points = 60 + (jump_distance - kpoint) * meter_value;
end
```

b) (10 %) Et skihopp belønnes med 0–60 stilpoeng. Fem dommere gir 0–20 poeng hver. Den laveste og den høyeste poengsummen strykes, og de tre resterende poengsummene legges sammen og utgjør hoppets stilpoeng.

Hvis et skihopp får poengsummene 17, 17,5, 17,5, 18, 19, strykes 17 og 19, og stilpoengene blir 17,5 + 17,5 + 18 = 53.

Skriv en funksjon *style_points* som tar inn en usortert liste *points* med de fem dommerpoengsummene, og returnerer hoppets stilpoeng.

Alternativ 1:

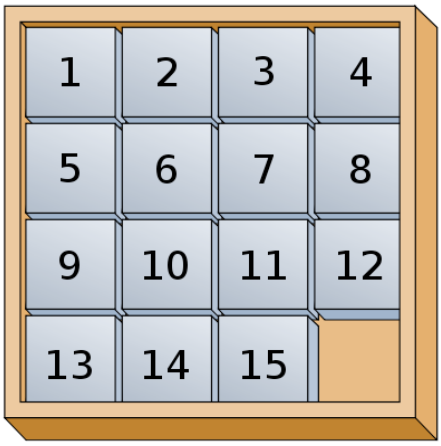
```
function points = style_points(refpoints)
      points = sum(refpoints) - min(refpoints) - max(refpoints);
end
```

Alternativ 2:

```
function points = style_points(points_list)
      points_list = sort(points_list);
      points = sum(points_list(2:4));
end
```

Oppgave 4: Programmering (50 %)

I denne oppgaven skal du programmere funksjoner til "15-spillet" som er illustrert i figur 1. Spillet består av et brett med 4x4 ruter med 15 brikker med tallene 1 til 15. En rute er tom, og den kan brukes til å flytte om på brikkene innbyrdes for å endre på rekkefølgen. Spillet starter ved at brikkene står i tilfeldig rekkefølge. Målet med spillet er å få brikkene i riktig rekkefølge fra 1 til 15 med den siste ruten fri slik som vist i figur 1. Spillebrettet skal i koden representeres som en 4x4-tabell med tall som vist i figur 2.



Figur 1. Spillebrett til "15-spillet".

	1	2	3	4
1	1	2	3	4
2	5	6	7	8
3	9	10	11	12
4	13	14	15	0

Figur 2. Spillebrettet representert som tabell.

I denne oppgaven er det hensiktsmessig å gjenbruke funksjoner du lager. Du kan *bruke* funksjoner fra andre deloppgaver selv om du ikke har klart å løse deloppgaven hvor du skal lage funksjonen.

a) (5 %) Skriv funksjonen *number_in_list* som tar inn et tall, *number,* og en liste (endimensjonal tabell) med tall, *list*. Hvis tallet *number* finnes i listen skal funksjonen returnere true, ellers false.

Alternativ 1:

```
function inlist = number_in_list(number, list)
      inlist = false;
      for i = 1:length(list)
            if list(i) == number
                  inlist = true;
                  return % return er ikke nødvendig
            end
      end
end
```

Alternativ 2:

```
function found = number_in_list(number, list)
      found = sum(list == number);
end
```

b) (5 %) Skriv funksjonen *random_list* som tar inn et tall, *number,* og returnerer en liste (endimensjonal tabell) med tallene fra og med 1 til og med tallet *number* i tilfeldig rekkefølge. Bruk funksjonen *number_in_list* fra oppgave a) i løsningen av denne oppgaven.

Gjør man funksjonskallet *random_list*(15) kan for eksempel følgende tabell returneres:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
3	2	15	13	12	9	1	11	5	14	7	6	4	8	10

Figur 3. Tabell med 15 tall i tilfeldig rekkefølge.

Alternativ 1:

```
function list = random_list(number)
      list = [];
      for i = 1:number
            n = ceil(rand() * number);
            while (number_in_list(n, list))
                  n = ceil(rand() * number);
            end
            list = [list n];
      end
end
```

Alternativ 2:

```
function list = random_list(number)
      num_drawn = 0;
      list = [];

      while (num_drawn < number)
```

```
            draw = ceil(number*rand);
            if (~number_in_list(draw, list)) % == false, ~ eller !
                  num_drawn = num_drawn + 1;
                  list(num_drawn) = draw;
            end
      end
end
```

c) (5 %) Skriv funksjonen *new_level* som tar inn en liste (endimensjonal tabell), *list*, bestående av 15 tall i tilfeldig rekkefølge som vist i figur 3. Funksjonen skal returnere en 4x4-tabell der tallene i *list* er satt inn fortløpende, radvis, fra rad 1, kolonne 1, til rad 4, kolonne 3. Elementet med indeks 4,4 skal ha verdien 0.

Kalles funksjonen *new_level* med listen vist i figur 3, skal den returnere tabellen vist i figur 4.

	1	2	3	4
1	3	2	15	13
2	12	9	1	11
3	5	14	7	6
4	4	8	10	0

Figur 4. Et tilfeldig spillebrett i 4x4-tabell.

Alternativ 1:

```
function level = new_level(list)
      level = zeros(4);
      for i = 1:4
            for j = 1:4
                  if (i < 4 || j < 4)
                        level(i,j) = list((i-1)*4 + j);
                  end
            end
      end
end
```

Alternativ 2:

```
function level = new_level(list)
      level = [list(1:4); list(5:8); list(9:12); list(13:15) 0];
end
```

Alternativ 3:

```
function level = new_level(list)
      % reshape is *not* in the function list
      level = reshape([list 0], 4, 4)';
end
```

d) (5 %) Skriv funksjonen *find_empty* som tar inn et spillebrett, *level*, (4x4-tabell, se figur 4) og returnerer posisjonen (rad, kolonne) til den tomme ruten (med verdi 0).

Alternativ 1:

```
function [rad kolonne] = find_empty(level)
      for i = 1:4
            for j = 1:4
                  if level(i,j) == 0
                        rad = i;
                        kolonne = j;
                  end
            end
      end
end
```

Alternativ 2:

```
function [row col] = find_empty(level)
      for row = 1:4
            for col = 1:4
                  if (level(row,col) == 0)
                        return
                  end
            end
      end
end
```

Alternativ 3:

```
function [row col] = find_empty2(level)
      [dummy col] = min(min(level));
      [dummy row] = min(min(level'));
end
```

e) (10 %) Skriv funksjonen *move_tile* som tar inn et spillebrett, *level*, og en tekststreng, *direction*, som angir retning der "l" er venstre (left), "r" er høyre (right), "d" er ned (down), og "u" er opp (up). Funksjonen skal returnere et spillebrett der ruten med tallet 0 har byttet plass med tallet som befinner seg i naboruten i angitt retning hvis et slikt bytte er mulig. Hvis byttet ikke er mulig skal funksjonen returnere et uendret spillebrett.

I eksemplet vist i figur 4, kan ruten med tallet 0 bytte plass opp (med tallet 6) og til venstre (med tallet 10).

Merk at funksjonen skal være generell og fungere uansett hvor ruten med tallet 0 er plassert i tabellen. Hvis funksjonen *move_tile* kalles med tabellen vist i figur 4 og retning "u" (opp) som parametre, så vil tallet 6 bytte plass med tallet 0 i tabellen. Bruk funksjonen *find_empty* fra oppgave d) i løsningen av denne oppgaven.

Alternativ 1:

```
function lvl = move_tile(level, direction)
      lvl = level;
      [r k] = find_empty(lvl);
      if direction == 'u'
            if r > 1
                  lvl(r,k) = lvl(r-1,k);
                  lvl(r-1,k) = 0;
            end
      elseif direction == 'd'
            if r < 4
                  lvl(r,k) = lvl(r+1,k);
                  lvl(r+1,k) = 0;
            end
      elseif direction == 'l'
            if k > 1
                  lvl(r,k) = lvl(r,k-1);
                  lvl(r,k-1) = 0;
            end
      elseif direction == 'r'
            if k < 4
                  lvl(r,k) = lvl(r,k+1);
                  lvl(r,k+1) = 0;
            end
      end
end
```

Alternativ 2:

```
function level = move_tile(level, direction)
      [r c] = find_empty(level);

      % map from udlr to +/- 1
      dr = 0; dc = 0;
      if (direction == 'u')
            dr = -1;
      elseif (direction == 'd')
            dr = 1;
      elseif (direction == 'l')
            dc = -1;
      elseif (direction == 'r')
            dc = 1;
      else
            return  % not required
      end

      % check for off-board movement
      if (r+dr < 1 || r+dr > 4 || c+dc < 1 || c+dc > 4)
            return
      end
```

```
      % move tile
      level(r,c) = level(r+dr, c+dc);
      level(r+dr, c+dc) = 0;
end
```

f) (5 %) Skriv funksjonen *correct_place* som tar inn et spillebrett, *level,* og returnerer antall tall som er riktig plassert på spillebrettet. Korrekt plassering av tallene er som vist i figur 2. Kalles funksjonen *correct_place* med spillebrettet fra figur 4, vil funksjonen gi svaret 2, ettersom tallene 0 og 2 er korrekt plassert på spillebrettet.

```
function count = correct_place(level)
      correct = [1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 0];
      count = 0;
      for i = 1:4
            for j = 1:4
                  if level(i,j) == correct((i-1)*4 + j)
                        count = count + 1;
                  end
            end
      end
end
```

g) (5 %) Skriv funksjonen *level_to_text* som tar inn et spillebrett, *level*, og skriver ut spillebrettet slik som vist i figur 5.

```
+----+----+----+----+
| 14 |  9 |  7 |  8 |
+----+----+----+----+
|  1 | 10 |  5 |  4 |
+----+----+----+----+
|  2 | 11 | 13 |  3 |
+----+----+----+----+
| 12 | 15 |  6 |  0 |
+----+----+----+----+
```

Figur 5. Spillebrett formatert som tekst.

Hint: Formatstrengen "%2d" brukes for å skrive ut et heltall, høyrejustert i et to tegn bredt felt.

```
function dummy = level_to_text(level)
      fprintf(1, '+----+----+----+----+\n');
      for i = 1:4
            fprintf(1, '| %2d | %2d | %2d | %2d |\n', level(i,1),
                    level(i,2), level(i,3), level(i,4));
            fprintf(1, '+----+----+----+----+\n');
      end
end
```

h) (10 %) Lag koden for å utføre følgende i et Matlab-skript:

1. Lag et spillebrett med tallene 1 til 15 i tilfeldig rekkefølge. Vi ønsker at spillebrettet skal ha minst 10 tall på riktig sted, så skriptet må lage nye brett helt til et slikt spillebrett er funnet.

2. Skriv ut spillebrettet med minst 10 riktig plasserte tall på skjermen ved hjelp av funksjonen *level_to_text*.

```
% 1.
level = zeros(4);
while (correct_place(level) < 10)
      level = new_level(random_list(15));
end

% 2.
level_to_text(level)
```

Svarskjema flervalgsoppgave

Studentnummer: _____ Linje: _____

Fagkode: _____ Dato: _____

Antall sider: _____ Side: _____

Oppgave nr.	*A*	*B*	*C*	*D*
1.1			X	
1.2				X
1.3	X			
1.4			X	
1.5			X	
1.6		X		
1.7	X			
1.8		X		
1.9		X		
1.10			X	
1.11			X	
1.12	X			
1.13			X	
1.14			X	
1.15		X		
1.16			X	
1.17	X			
1.18		X		
1.19			X	
1.20		X		
1.21			X	
1.22				X
1.23	X			
1.24			X	
1.25			X	