NTNU
Norges teknisk-naturvitenskapelige universitet

Fakultetet for informasjonsteknologi, matematikk og elektroteknikk

Institutt for datateknikk og informasjonsvitenskap

BOKMÅL

Sensurfrist: 11. januar 2013

Avsluttende eksamen i TDT4105 Informasjonsteknologi, grunnkurs

Tirsdag 11. desember 2012
9:00 – 13:00

Faglig kontakt under eksamen:

Roger Midtstraum	tlf.	995 72 420
Rune Sætre	tlf.	452 18 103
Alf Inge Wang	tlf.	922 89 577
Erik Smistad	tlf.	905 24 585

Hjelpemidler: C

Typegodkjent kalkulator: HP30S

Sensur:

Resultater gjøres kjent på 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.

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 kjennetegner komprimeringsalgoritmer som er tapsløs (lossless)?
   a) Den opprinnelige datamengden kan gjenskapes nøyaktig.
   b) Den komprimerte datamengden er like stor som utgangspunktet.
   c) Egner seg spesielt godt for multimediedata som bilder, lyd og video.
   d) Fjerner bare informasjonsinnhold som ikke er viktig for menneskers oppfatning av informasjonsmengden, for eksempel i et bilde.

2) Hva sier Nyquist-regelen om samplingsfrekvensen?
   a) Samplingsfrekvensen må være minst halvparten av den høyeste lydfrekvensen.
   b) Samplingsfrekvensen må være minst den samme som den høyeste lydfrekvensen.
   c) Samplingsfrekvensen må være minst dobbelt så rask som den høyeste lydfrekvensen.
   d) Samplingsfrekvensen må alltid være 20KHz.

3) Hva er oppgaven til en programteller (program counter):
   a) Den holder rede på antall kodelinjer i et program.
   b) Den inneholder adressen til neste instruksjon.
   c) Den styrer antall iterasjoner i en FOR-løkke.
   d) Ingen av svarene er riktige.

4) Hvor mange symboler kan kodes med 10 bit?
   a) 10.
   b) 512.
   c) 1024.
   d) Ingen.

5) Hva er fokuset i programvarevalideringsfasen i systemutvikling?
   a) Beskrive hva systemet skal gjøre.
   b) Designe hvordan systemet skal oppføre seg.
   c) Teste om systemet stemmer med spesifikasjonen og kundekrav.
   d) Ingen av svarene er riktige.

6) Hvorfor digitalisere nettverk?
   a) Ønske om å kombinere tjenester.
   b) Enklere å utnytte kapasitet bedre med felles nettverk.
   c) Digital koding kan gi bedre feilsjekk og korrigering av feil.
   d) Alle alternativene a-c er riktige.

7) Hva er et Denial of Service–angrep?

a) Å sende så mange forespørsler til en tjener (server) at den ikke klarer å utføre oppgavene sine.
b) Å bryte seg inn på en tjener (server) og sørge for at den nekter å utføre tjenestene sine.
c) Å nekte å motta meldinger fra en tjener (server) som da blir opptatt med å sende forespørslene på nytt og på nytt.
d) Å sende en falsk e-post om problemer med en tjeneste, som for eksempel en nettbank, og lure brukeren til å avsløre påloggingsinformasjon for å få løst problemet.

8) Hva vil det si at vi at vi har "random access" (tilfeldig tilgang) til minnet?

a) All data i minnet kan hentes direkte uansett hvor det befinner seg.
b) Det er tilfeldig hva som hentes ut av minnet.
c) Vi må hente ut data sekvensielt (byte for byte) for å finne det vi leter etter.
d) Ingen av svarene er riktige.

9) Hva vil det si at en datamaskin er deterministisk?

a) Den har en pessimistisk livsanskuelse som avviser fri vilje.
b) Når den skal velge hvilken instruksjon den skal behandle neste gang har den ikke noe valg, men baserer valget på programmet og dataene den gis.
c) At den har en intuisjon på hva som er lurt å gjøre.
d) Ingen av svarene er riktige.

10) Hvilke av alternativene under er mulige tolkninger av PandA-mønstre?

a) True og False.
b) Ja og Nei.
c) + og –.
d) Alle alternativene a-c er riktige.

11) Når vi studerer algoritmers effektivitet, ser vi på hvordan kjøretiden utvikler seg i forhold til mengden av input. Vi gjør analyser av bestefall, verstefall og gjennomsnittstilfellet. Hvorfor er det spesielt interessant å analysere en algoritmes kjøretid i verste fall?

a) Det setter en øvre grense for hvor lang tid det tar å kjøre algoritmen.
b) Programmerere er pessimister.
c) Det er mer interessant med høye tall.
d) Det forteller hvor lang tid algoritmen ca. bruker på å kjøre.

12) Vi deler inn algoritmer i klasser basert på de funksjoner som beskriver deres utvikling i kjøretid best. Hvilken algoritme er i klassen Θ(log(n)):

a) Innstikksortering (Insertion sort).
b) Binærsøk (binary search).
c) Sekvensielt søk (sequential search).
d) Ingen av svarene er riktige.

13) Hva er et datagram?

a) Et telegram som er skrevet på data.
b) Vekten på en dataenhet.
c) En pakke som sendes over internett som følger IP-protokollen.
d) Ingen av svarene er riktige.

14) Fargene som vises på en dataskjerm representeres ofte med 24 bits RGB-koding. Fargen blå vil da representeres som:

a) 0000 0000 0000 0000 0000 0000.
b) 1111 1111 0000 0000 0000 0000.
c) 0000 0000 1111 1111 0000 0000.
d) 0000 0000 0000 0000 1111 1111.

15) Når vi overfører data over internett oppstår det ofte feil på grunn av forstyrrelser på linjene. For å oppdage slike feil brukes ofte

a) NIC (Network Interface Card).
b) ISP (Internet Service Provider).
c) CRC (Cyclic Redundancy Check).
d) Ingen av svarene er riktige.

16) En mikroprosessor utfører de samme fem oppgavene om og om igjen. Hvilken rekkefølge av stegene under beskriver korrekt rekkefølge på dette F/E-kretsløpet (F/E cycle)?

a) Information Fetch – Data Fetch – Instruction Decode – Instruction Execution – Results Return.
b) Results Return – Instruction Execution – Information Fetch – Data Fetch – Instruction Decode.
c) Information Fetch – Instruction Decode – Data Fetch – Instruction Execution – Results Return.
d) Instruction Decode – Instruction Execution – Information Fetch – Data Fetch – Results Return.

17) Hvilken bestemt endring har vi sett de siste årene innen systemutvikling?

a) Spesifisering av krav er ikke lengre relevant.
b) Smidig (agile) systemutvikling har overtatt mer og mer for plandrevet systemutvikling.
c) Vannfallsmodellen har overtatt for inkrementell systemutvikling.
d) Ingen av svarene er riktige.

18) Hva er DAC?

a) Et program som hjelper med beregninger (data-assisted computing).
b) En enhet som oversetter analoge signaler til digitale signaler.
c) En enhet som oversetter digitale signaler til analoge signaler.
d) Et program som oversetter datakode til programmeringsspråket C (evt. C++).

19) Rekursjon betyr at

a) En funksjon kaller seg selv.
b) Kjøretiden til programmet minsker.
c) Programmet går i evig løkke.
d) Ingen av svarene er riktige.

20) Ranger effektivitetsklassene $\Theta(n^3)$, $\Theta(n)$, $\Theta(\log(n))$ og $\Theta(n^2)$ etter effektivitet, der minst effektiv først og der etter mer og mer effektiv

a) $\Theta(n^3)$, $\Theta(n^2)$, $\Theta(n)$, $\Theta(\log(n))$.
b) $\Theta(n^3)$, $\Theta(n)$, $\Theta(n^2)$, $\Theta(\log(n))$.
c) $\Theta(\log(n))$, $\Theta(n^3)$, $\Theta(n^2)$, $\Theta(n)$.
d) $\Theta(n)$, $\Theta(\log(n))$, $\Theta(n^3)$, $\Theta(n^2)$.

Oppgave 2 – Grunnleggende programmering (20 %)

Du kan anta at alle funksjonene mottar gyldige input-verdier.

Oppgave 2 a) (5 %)
Lag funksjonen **`summerOlympics`** som har inn-parametere **`firstYear`** og **`lastYear`**. Funksjonen skal returnere parameteren **`years`**, som er en vektor med alle OL-årene fra og med **`firstYear`** til og med **`lastYear`** (inkludert framtidige planlagte år for sommer-OL). Fra og med OL i London i 1948 har sommer-OL vært arrangert hvert fjerde år.

Du kan anta at **`firstYear`** ≥ 1948.

Eksempel på kjøring av funksjonen og hva den returnerer:

```
>> summerOlympics(1999,2012)
ans =
     2000      2004      2008      2012
>>
```

Oppgave 2 b) (7,5 %)
Lag funksjonen **`findAge`** som har inn-parametere **`bYear`**, **`bMonth`**, **`bDay`** som er tre heltall som beskriver dato for en fødselsdag. Funksjonen skal returnere **`age`** som beskriver hvor gammel en person med oppgitt fødselsdag (**`bYear`**, **`bMonth`** og **`bDay`**) er i dag angitt i hele år.

For å finne år, måned og dag for i dag ***skal du bruke*** en eksisterende funksjon som heter **`current_date()`**. Funksjonen returnerer tre heltall på formatet **`[yyyy, mm, dd]`**.

Eksempel på bruk av funksjonen **`current_date`**:

`[yyyy, mm, dd] = current_date()` gir i dag `yyyy=2012`, `mm=12`, `dd=11`.

Eksempel på kjøring av funksjonen **`findAge`** og hva den returnerer:

```
>> findAge(2000,12,15)
ans =
     11
>>
```

Oppgave 2 c) (7,5 %)
Lag en funksjon **`printAgeDiff`** som tar imot en vektor av *structer* som beskriver personer med følgende felter: **`fName`**, **`lName`**, **`bYear`**, **`bMonth`**, **`bDay`**. Funksjonen skal bruke funksjonen **`findAge`** fra oppgave 2b (kan bruke funksjonen selv om du ikke har løst oppgave 2b) til å sammenlikne alderen i hele år på etterfølgende personer i vektoren og gjøre følgende:

- Hvis person n og person n+1 har samme alder angitt i antall hele år, skal følgende skrives ut til skjerm:
  <fName n> <lName n> is at the same age as <fName n+1> <lName n+1>
- Hvis person n er eldre enn person n+1 angitt i antall hele år, skal følgende skrives ut til skjerm:
  <fName n> <lName n> is older than <fName n+1> <lName n+1>
- Hvis person n er yngre enn person n+1 angitt i antall hele år, skal følgende skrives ut til skjerm:
  <fName n> <lName n> is younger than <fName n+1> <lName n+1>

Eksempel på *structer* som beskriver fire kjente personer:

```
bieber  = struct('fName','Justin','lName','Bieber','bYear',1994,'bMonth',3,'bDay',1);
donald = struct('fName','Donald','lName','Duck','bYear',1934,'bMonth',8,'bDay',1);
george = struct('fName','George','lName','Clooney','bYear',1961,'bMonth',5,'bDay',6);
eddie   = struct('fName','Eddie','lName','Murphy','bYear',1961,'bMonth',4,'bDay',3);
```

Eksempel på kjøring av funksjonen **`printAgeDiff`** med *structene* bieber, donald, george og eddie:

```
>> printAgeDiff([bieber, donald, george, eddie])
Justin Bieber is younger than Donald Duck
Donald Duck is older than George Clooney
George Clooney is at the same age as Eddie Murphy
>>
```

Oppgave 3 – Kodeforståelse (15 %)

Oppgave 3 a) (5 %)
Hva returneres hvis funksjonen **`fu1(1234)`** med kode som vist under kjøres?

```
function r = fu1(a)
    r = 0;
    while a > 0
        s = rem(a,10);
        r = r+s;
        a = (a-s)/10;
    end % while

end % function
```

Oppgave 3 b) (5 %)
Hva blir verdiene til a, b, c og d etter kallet
`[a b c d] = fu2('Ut på tur, aldri sur')`
med koden som vist under?

```
function [r s t u] = fu2(input)
    r = 0;
    s = 0;
    t = 0;
    u = 0;
    n = length(input);

    for i = 1:n
        switch lower(input(i))
            case {'a','b','c','d','e','f','g',...
                  'h','i','j','k','l','m','n',...
                  'o','p','q','r','s','t','u',...
                  'v','w','x','y','z','æ','ø','å'}
                r = r + 1;
            case {'1','2','3','4','5','6','7','8','9','0'}
                s = s + 1;
            case {' '}
                t = t + 1;
            otherwise
                u = u + 1;
        end % switch
    end % for

    r = 100*r/n;
    s = 100*s/n;
    t = 100*t/n;
    u = 100*u/n;

end % function
```

Oppgave 3 c) (5 %)
Hva returneres av kallet **`fu3(100)`** med koden som vist under?

```
function r = fu3(a)

    if a <= 2
        r = 1;
    else
        r = 1 + fu3(a/2);
    end % if

end % function
```

Oppgave 4 – Programmering (40 %)

Denne oppgaven fokuserer på behandling av data fra fire værsensorer som måler en verdi per døgn av følgende data:

- Temperatur: Angis som heltall i Celsius fra -50 C til + 50 C
- Nedbør: Angis som heltall i mm nedbør per døgn fra 0 til 2000 mm
- Luftfuktighet. Angis som heltall fra 0 til 100 %
- Vindstyrke: Angis som heltall fra 0 til 50 meter per sekund

Hvis ikke noe annet er oppgitt kan du anta korrekt input til funksjonene.

Oppgave 4 a) (5 %)
Lag en funksjon **`cold_days`** som tar imot parameteren **`templist`**, som en liste av temperaturer, og returnerer parameteren **`days`**, som angir antall døgn der temperaturen var under 0 grader.

Eksempel på kall av funksjonen og hva den returnerer:

```
>> cold_days([1,-5,3,0,-6,-3,15,0])
ans =
  3
```

Oppgave 4 b) (5 %)
Lag en funksjon **`cap_data`** som har inn-parameterne **`array`** (tabell med data), **`min_value`** (minimumsverdi) og **`max_value`** (maksimumsverdi). Funksjonen skal returnere tabellen **`result`** der alle elementer i tabellen **`array`** som har verdi mindre enn **`min_value`** skal settes lik **`min_value`** og alle elementer i tabellen som har verdi høyere enn **`max_value`** skal settes lik **`max_value`**.

Eksempel på kall av funksjonen og hva den returnerer (endrede verdier i fet skrift):

```
>> A = [-70,30,0,90,23,-12,95,12];
>> cap_data(A,-50,50)
ans =
 -50 30 0 50 23 -12 50 12
>>
```

Oppgave 4 c) (10 %)
Lag en funksjon **`generate_testdata`** som har inn-parameterne **`N`**, **`min_value`** (minimumsverdi) og **`max_value`** (maksimumsverdi). Funksjonen skal returnere tabellen **`result`** som består av **`N`** unike *tall* (heltall) som blir trukket tilfeldig der {**`min_value`** $\leq$ *tall* $\leq$ **`max_value`**}. Unik betyr her at ingen elementer i tabellen **`result`** skal ha samme verdi. Du kan anta at antall mulige verdier i intervallet tallet blir trukket fra alltid vil være større enn **`N`**.

Eksempel på kall av funksjonen og hva den returnerer:

```
>> temperatures = generate_testdata(10,-5,10)
temperatures
 -5 3 7 9 -3 4 2 0 -1 5
>>
```

Oppgave 4 d) (5 %)
Lag en funksjon **`create_db`** som har inn-parameterne **`temp`**, **`rain`**, **`humidity`** og **`wind`**, som er fire tabeller av samme størrelse (likt antall elementer) med data for temperatur, nedbør, luftfuktighet og vind.

Funksjonen skal lage og returnere en liste (vector) med strukturer (structures) som inneholder alle dataene fra de fire inn-parameterne. Hver struktur skal ha feltene (fields) temp, rain, humidity og wind, og inneholde væredata for en bestemt dag. Strukturen med indeks 1 i listen skal inneholde væredata for dag 1, strukturen med indeks 2 i listen skal inneholde værdata for dag 2 og så videre.

Eksempel på kall av funksjonen og hva den returnerer:

```
>> temp = [1,5,3]
>> rain = [0,30,120]
>> humidity = [30,50,65]
>> wind = [5,3,7]
>> db = create_db(temp, rain, humidity, wind)
db =
1x3 struct array with fields:
  temp
  rain
  humidity
  wind
>> db(1)
ans =
    temp: 1
    rain: 0
  humidity: 30
    wind: 5
>> db(2)
ans =
    temp: 5
    rain: 30
  humidity: 50
    wind: 3
>> db(3)
ans =
    temp: 3
    rain: 120
  humidity: 65
    wind: 7
>>
```

Oppgave 4 e) (5 %)
Lag en funksjon **`print_db`** som har inn-parameteren **`weather`**, som er en liste med strukturer som beskrevet i oppgave 4d. Funksjonen skal skrive ut innholdet i **`weather`** på skjerm etter følgende format og med overskrift som vist på utskriften nederst i deloppgaven:

- Day (dag) – høyrejustert med 4 tegn
- Temp (temperatur) – høyrejustert med 6 tegn
- Rain (nedbør) – høyrejustert med 6 tegn
- Humidity (luftfuktighet) – høyrejustert med 10 tegn
- Wind (vind) – høyrejustert med 6 tegn

Eksempel på kall av funksjonen ved bruk av listen fra oppgave 4d:

```
>> print_db(db)

Day | Temp | rain | humidity | wind
====+======+======+==========+======
   1      1      0         30      5
   2      5     30         50      3
   3      3    120         65      7
>>
```

Oppgave 4 f) (10 %)
Lag funksjonen **`strange_weather`** som har inn-parameterne **`temp`** og **`rain`**, som er to tabeller med data for temperaturer og regn av lik størrelse (samme antall elementer).

Funksjonen skal returnere **`start`** (startdag) og **`stop`** (sluttdag) for det lengste intervallet der det er minusgrader, samt at temperaturen faller samtidig som nedbørsmengden stiger i etterfølgende dager. Indeksering av dager starter på 1. Hvis ingen etterfølgende dager har denne karakteristikken, returneres (0,0).

Eksempel på kall av funksjonen (med intervall som oppfyller kravet uthevet):

```
>> temp=[1, 3, 4,-5,-6,-7,-8,-9,3,0]
>> rain=[0,20,30, 0,10,30,50, 0,5,2]
>> [start,stop] = strange_weather(temp,rain)
start =
     4
stop =
     7
>>
```

Appendiks: Nyttige funksjoner i Matlab

FIX Round towards zero.
FIX(X) rounds the elements of X to the nearest integers towards zero.

FLOOR Round towards minus infinity.
FLOOR(X) rounds the elements of X to the nearest integers towards minus infinity.

FCLOSE Close file.
ST = FCLOSE(FID) closes the file associated with file identifier FID, which is an integer value obtained from an earlier call to FOPEN. FCLOSE returns 0 if successful or -1 if not.

FEOF Test for end-of-file.
ST = FEOF(FID) returns 1 if the end-of-file indicator for the file with file identifier FID has been set, and 0 otherwise. The end-of-file indicator is set when a read operation on the file associated with the FID attempts to read past the end of the file.

FGETL Read line from file, discard newline character.
TLINE = FGETL(FID) returns the next line of a file associated with file identifier FID as a MATLAB string. The line terminator is NOT included. Use FGETS to get the next line with the line terminator INCLUDED. If just an end-of-file is encountered, -1 is returned.

FOPEN Open file.
FID = FOPEN(FILENAME,PERMISSION) opens the file FILENAME in the mode specified by PERMISSION:

'r' open file for reading
'w' open file for writing; discard existing contents
'a' open or create file for writing; append data to end of file
'r+' open (do not create) file for reading and writing
'w+' open or create file for reading and writing; discard existing contents
'a+' open or create file for reading and writing; append data to end of file

FPRINTF Write formatted data to file.
COUNT = FPRINTF(FID,FORMAT,A,...) formats the data in the real part of array A (and in any additional array arguments), under control of the specified FORMAT string, and writes it to the file associated with file identifier FID. COUNT is the number of bytes successfully written. FID is an integer file identifier obtained from FOPEN. It can also be 1 for standard output (the screen) or 2 for standard error. If FID is omitted, output goes to the screen.

FORMAT is a string containing ordinary characters and/or C language conversion specifications. Conversion specifications involve the character %, optional flags, optional width and precision fields, optional subtype specifier, and conversion characters d, i, o, u, x, X, f, e, E, g, G, c, and s.

The special formats \n,\r,\t,\b,\f can be used to produce linefeed, carriage return, tab, backspace, and formfeed characters respectively. Use \\ to produce a backslash character and %% to produce the percent character.

LENGTH Length of vector.
LENGTH(X) returns the length of vector X. It is equivalent to MAX(SIZE(X)) for non-empty arrays and 0 for empty ones.

LOWER Convert string to lowercase.
B = lower(A) converts any uppercase characters in A to the corresponding lowercase character and leaves all other characters unchanged.

MOD Modulus after division.
MOD(x,y) is x - n.*y where n = floor(x./y) if y ~= 0.

RAND Uniformly distributed pseudorandom numbers.
R = RAND(N) returns an N-by-N matrix containing pseudorandom values drawn from the standard uniform distribution on the open interval(0,1). RAND(M,N) or RAND([M,N]) returns an M-by-N matrix.

RANDI Pseudorandom integers from a uniform discrete distribution.
R = RANDI(IMAX,N) returns an N-by-N matrix containing pseudorandom integer values drawn from the discrete uniform distribution on 1:IMAX.
RANDI(IMAX,M,N) or RANDI(IMAX,[M,N]) returns an M-by-N matrix.

REM Remainder after division.
REM(x,y) is x - n.*y where n = fix(x./y) if y ~= 0.

SIZE Size of array.
D = SIZE(X), for M-by-N matrix X, returns the two-element row vector
D = [M,N] containing the number of rows and columns in the matrix.

SQRT Square root.
SQRT(X) is the square root of the elements of X.

STR2NUM Convert string matrix to numeric array.
X = STR2NUM(S) converts a character array representation of a matrix of numbers to a numeric matrix. For example,
S = ['1 2' ; '3 4'] str2num(S) => [1 2;3 4]

SUM Sum of elements.
S = SUM(X) is the sum of the elements of the vector X. If X is a matrix, S is a row vector with the sum over each column.

UPPER Convert string to uppercase.
B = upper(A) converts any lower case characters in A to the corresponding upper case character and leaves all other characters unchanged.

Svarskjema flervalgsoppgave

Kandidatnummer: _____ Program: _____

Fagkode: _____ Dato: _____

Antall sider: _____ Side: _____

Oppgavenr	*A*	*B*	*C*	*D*
1.1				
1.2				
1.3				
1.4				
1.5				
1.6				
1.7				
1.8				
1.9				
1.10				
1.11				
1.12				
1.13				
1.14				
1.15				
1.16				
1.17				
1.18				
1.19				
1.20				

Svarskjema flervalgsoppgave

Kandidatnummer: _____ Program: _____

Fagkode: _____ Dato: _____

Antall sider: _____ Side: _____

Oppgavenr	*A*	*B*	*C*	*D*
1.1				
1.2				
1.3				
1.4				
1.5				
1.6				
1.7				
1.8				
1.9				
1.10				
1.11				
1.12				
1.13				
1.14				
1.15				
1.16				
1.17				
1.18				
1.19				
1.20				