NTNU

Fakultet for informasjonsteknologi,
matematikk og elektroteknikk

Institutt for datateknikk
og informasjonsvitenskap

BOKMÅL

KONTINUASJONSEKSAMEN TDT 4105

Informasjonsteknologi, grunnkurs

Tirsdag 10. august 2010, 9.00–13.00

Faglig kontakt under eksamen:
Jørn Amundsen (918 97 897)

Hjelpemidler (C):
Tilleggshefte I, "Introduksjon til HTML, CSS, JSP og MYSQL" (alle utgaver)
Tilleggshefte II, "introduksjon til: MATLAB" (alle utgaver)
Typegodkjent elementær kalkulator, enten HP 30S eller Citizen SR-270X.
Det er tillatt å stryke over med markeringspenn, men ikke å skrive i tilleggsheftene.

Sensurdato:
31. august 2010. Resultater gjøres kjent på http://studweb.ntnu.no.

Oppgavesettet inneholder 3 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!

Oppgave 1 – Flervalgsoppgaver (20 %)

Bruk vedlagt svarskjema for å svare på denne oppgaven. 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. Hvor mange biter (bits) trenger du til å kode det norske alfabetet (29 forskjellige bokstaver) ved enkel koding hvor alle symbolene er representert med samme antall biter ?

   a) 5 biter
   b) 6 biter
   c) 8 biter

2. Kalle har klart å knekke passordet til en database som lagrer alle karakterene på NTNU etter flere forsøk, men ombestemmer seg og ser ikke på innholdet i databasen. Har Kalle gjort noe ulovlig i følge loven ?

   a) Nei
   b) Ja
   c) Loven sier ikke noe om dette

3. Hvilken type minne har normalt høyest tilgangstid (access time) i en datamaskin ?

   a) Registre
   b) Cache
   c) Primærminne

4. En ulempe med åpen kildekode (open source) er:

   a) Du kan ikke endre produktet selv
   b) Det er store firma står som regel bak utviklingen
   c) Du har normalt ikke krav på brukerstøtte

5. Hva er beskriver en kravspesifikasjon ?

   a) Kravspesifikasjon beskriver HVA systemet skal gjøre
   b) Kravspesifikasjon beskriver HVORDAN systemet skal lages
   c) Kravspesifikasjon beskriver koden til et system

6. Hva er svart-boks testing ?

   a) Testing som konsentrerer seg om systemets eller modulens indre oppbygning
   b) Testing som ikke ser på systemets eller modulens indre oppbygning
   c) Testing som konsentrerer seg om hvordan de ulike delene i et system fungerer sammen

7. Hva er kjernen i et operativsystem ?

    a) Den delen som tar i mot input fra brukeren
    b) Den delen som administrerer maskinressurser
    c) Det mediet som operativsystemet installeres fra

8. Hvorfor krypteres data ?

    a) For at data ikke skal kunne leses av uvedkommende
    b) For at det skal ta mindre plass
    c) For at det skal kunne gjøres tilgjengelig for alle på Internett

9. En protokoll er:

    a) Et program som oversetter kildekode til maskinkode slik at det kan kjøres på en datamaskin
    b) En enhet som gjør beregninger og sammenlikninger i en datamaskin
    c) Et sett av kommunikasjonsregler

10. Er binærsøk alltid raskere enn sekvensielt søk ?

    a) Binærsøk er alltid raskere enn sekvensielt søk uansett
    b) Binærsøk er alltid raskere enn sekvensielt søk på sorterte datamengder
    c) Sekvensielt søk kan være raskere enn binærsøk

11. Hva gjør man når man designer et system under systemutvikling ?

    a) Beskriver hva systemet skal gjøre
    b) Beskriver hvordan systemet skal lages
    c) Beskriver kun hvordan systemet skal se ut

12. Hva er hovedoppgaven til en navnetjener ?

    a) Oversetter domenenavn til IP-adresser
    b) Oversetter IP-adresser til domenenavn
    c) Oversetter IP-adresser til URL

13. I det binære tallsystemet, hva er 010_2 multiplisert med 101_2 ?

    a) 111_2
    b) 1010_2
    c) 1111_2

14. Hva er inspeksjon i systemutvikling ?

    a) Inspeksjon innebærer å finne feil eller forsikre seg om at feil ikke finnes uten å kjøre programmet
    b) Inspeksjon innebærer å undersøke om de som programmerer lager ryddige programmer som er lette å lese
    c) Inspeksjon innebærer å finne feil i koden ved å kjøre programmet

15. Hva er en normal konfigurasjon av en klient-tjener arkitektur ?

    a) Få tjenere, mange klienter
    b) Mange tjenere, få klienter
    c) Mange tjenere, mange klienter

16. En bit av et program består av ei for-løkke inne i ei for-løkke. Begge løkkene løper igjennom tallverdiene 1:N. Tidsforbruket til denne programbiten vil være proporsjonalt med

    a) N
    b) $N \log N$
    c) N^2

17. Gitt ei sortert liste av 272 verdier. Vi søker etter en verdi som ikke finnes i lista. Hvor mange sammenligninger må vi i verste fall gjøre ved bruk av binærsøk ?

    a) 136
    b) 8
    c) 9

18. Hva er en database i følge læreboka ?

    a) En samling strukturerte data
    b) Et program for å håndtere store datamengder
    c) Informasjon + metainformasjon

19. Hvilken av følgende alternativer blir **ikke** i følge loven ansett som sensitive personopplysninger ved opprettelse av personregistre i følge læreboka ?

    a) Religiøs bakgrunn
    b) Karakterer
    c) Fagforeningsbakgrunn

20. Hva blir Huffmann-koden til symbolene A-G når frekvensfordelingen i en melding på 100 symboler er som vist i tabell 1 ?

    a) A:1, B:01, C:011, D:001, E:0011, F:0001, G:00011
    b) A:1, B:011, C:010, D:001, E:0001, F:00001, G:00000
    c) A:1, B:01, C:010, D:001, E:0010, F:0001, G:0010

Symbol	A	B	C	D	E	F	G
Antall	40	20	15	10	8	5	2

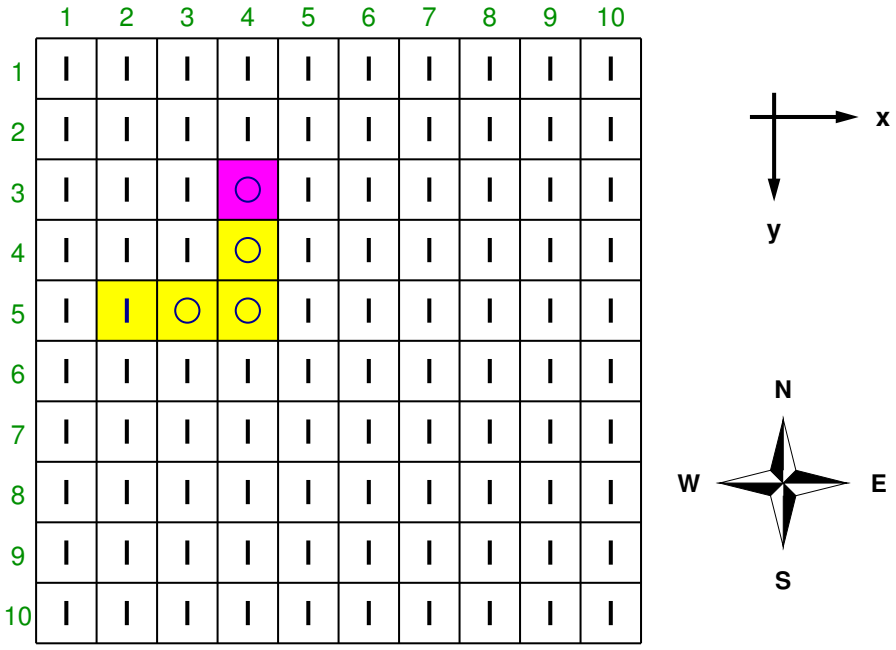
Tabell 1: Frekvensfordeling av symbolene A-G i en melding på 100 symboler

Oppgave 2 – Matlab programmering (70 %)

*Se vedlegget "List of built-in functions". I hver deloppgave er det angitt dersom det **ikke** er tillatt å bruke innebygde funksjoner, eller om navngitte innebygde funksjoner ikke er tillatt for å løse problemet. Hvis det ikke er tatt noen forbehold kan du velge fritt fra funksjonene i vedlegget.*

Bruk kun tallet `1` *eller nøkkelordet* `true` *for å angi en logisk* **sann** *verdi, og tilsvarende tallet* `0` *eller nøkkelordet* `false` *for en logisk* **usann** *verdi.*

I denne oppgaven skal du programmere funksjoner til et Snake-spill, der spilleren skal styre en slange som beveger seg rundt på et spillbrett. Spillbrettet en kvadratisk tabell av vilkårlig størrelse N, der "1" indikerer en *tom rute*. En slange av lengde L representeres ved to en-dimensjonale tabeller, en for hver akse (kolonner x og rader y). Figur 1 viser en slange av lenge $L = 5$ på et spillbrett av størrelse $N = 10$.



Figur 1: Slange av lengde 5 på et spillbrett av størrelse 10.

Tabellene for slangen i figur 1 er

```
snake_x = [4   4   4   3   2]
snake_y = [3   4   5   5   5]
```

Elementene er ordnet fra hodet til halen. Hodet ligger i punktet $(4, 3)$ og halen i $(2, 5)$. Merk at halen er tegnet inn som en "1", slik at slangen automatisk visker ut etter seg hvis den beveger seg over brettet.

a) (3%) Lag funksjonen `init_board` som tar inn størrelsen på spillbrettet som parameter N, og som returnerer et tomt spillbrett i form av en kvadratisk tabell hvor alle elementene har verdien **1**.

b) (10%) Lag en funksjon `feed_snake` som tar inn et spillbrett og legger til m døde mus i tilfeldige, tomme, punkt på brettet. Sett verdien av ruten til **5** for å angi at det ligger en mus der. Returner det oppdaterte spillbrettet. Bruk den innebygde funksjonen `rand` til å velge ut tilfeldige posisjoner på brettet.

c) (5%) Skriv funksjonen `is_food` som tar inn en x- og y-posisjon (slangehodet), og et spillbrett. Returner **sant** hvis det er mat i den angitte posisjonen, ellers **usant**. I hvilken dimensjon lagres x-posisjonene dersom spillbrettet lagres som en to-dimensjonal MATLAB-tabell ?

d) (10%) Skriv funksjonen `draw_snake` som tar inn et spillbrett og en slange av vilkårlig størrelse. Spillbrettet er representert ved en kvadratisk tabell og slangen ved to lister av hhv. x- og y-posisjoner. Anta begge listene er av samme lengde $L > 0$ uten å sjekke dette. Returner et oppdatert spillbrett, med 0'er i slangens posisjon, unntatt for halen som skal settes til 1.

Se bort fra eventuelle døde mus i rutene slangen skal plasseres.

e) (15%) Skriv funksjonen `valid_move` som tar inn lister for slangens posisjon (`snake_x` og `snake_y`) og en inngangsparameter for *retningen* slangen skal bevege seg, gitt ved en av bokstavene `'N'`, `'E'`, `'S'` og `'W'`. Returner **sant** hvis et flytt i angitt retning kan gjennomføres uten at slangen krasjer med seg selv, og **usant** hvis dette ikke er mulig eller hvis retningsparameteren har en ugyldig verdi.

Det er ikke nødvendig å sjekke om flyttet går til en ulovlig posisjon.

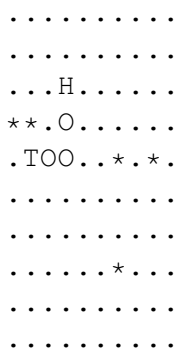
f) (12%) Skriv funksjonen `move_snake` som tar inn to lister for hhv. slangens x- og y-posisjoner (`snake_x` og `snake_y`), samt retningen slangen skal bevege seg (`'N'`, `'E'`, `'S'` eller `'W'`). Returner *oppdatere posisjoner* i de to listene, basert på verdien av parameteren for flyttretning.

Eksempel:

```
>> snake_x = [4  4  4  3  2];
>> snake_y = [3  4  5  5  5];
>> [snake_x snake_y] = move_snake(snake_x, snake_y, 'N')
snake_x =
   4   4   4   4   3
snake_y =
   2   3   4   5   5
```

g) (15%) Lag en funksjon `print_board` som skriver ut et spillbrett med slange og mat i kommandovinduet, grafisk i to dimensjoner ved hjelp av bokstaver og tegn fra tastaturet. Dette kalles *ASCII art* på engelsk. Bruk tegnene `'.'` for en tom rute, `'*'` for mat, `'O'` (stor bokstav) for slangens kropp, `'H'` for slangens hode og `'T'` for slangens hale. Ugyldige verdier behandles som tomme ruter. Se eksempel på utskrift av slangen fra figur 1 med 5 enheter mat i figur 2.

(oppgaven fortsetter på neste side)



Figur 2: Spillbrett med slange (HOOOT) og mat ($\star$) i tekst-grafikk.

Oppgave 3 – HTML (10 %)

I denne oppgaven får du oppgitt en HTML-fil med 10 mangler. Du skal skrive HTML-koden for å lage en webside som vist i figur 3.

Merk følgende opplysninger:

1. Alle innslag i tabellen er i utgangspunktet (default) venstrejustert horisontalt og midtjustert vertikalt.
2. Bildet `geek.jpg` skal brukes i venstre kolonne, og er 310 x 310 piksler.
3. Bildet `header.jpg` skal brukes i midtkolonnen, og er 825 x 105 piksler.
4. Venstre kolonne skal ha bredde på 120 piksler
5. Header i midtkolonnen skal ha høyde på 70 piksler.

Skriv HTML-kode for de tomme feltene merket **"Fyll inn"**.

Figur 3: HTML-fil vist i nettleser. Merk at grafikk som kan identifisere nettleseren er tatt bort.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN">
<html>
<head>
<meta http-equiv="Content-type" content="text/html;charset=iso-8859-1">
<title>Hjemmeside layout</title>
</head>
<body>
Fyll inn 1
<tr>
Fyll inn 2
Fyll inn 3
<h3>Meny</h3>
<a href="http://www.vg.no">Verdens Gang</a><br>
<a href="http://www.dn.no">Dagens Næringsliv</a><br>
<a href="http://www.vl.no">Vårt Land</a><br>
<a href="http://www.dagbladet.no">Dagbladet</a><br>
<a href="http://www.nrk.no">NRK</a><br>
</td>
<td>
Fyll inn 4
</td>
Fyll inn 5
Dato: 25.mai 2010
</td>
Fyll inn 6
<tr>
Fyll inn 7
<h1>Velkommen til hjemmesiden min!</h1>
<p>Her skal hoveddelen av websiden være.<br>
Her kommer det mye rart...<br></p>
<p>Lurer på om studentene i ITGK klarer layouten...<br>
Håper det...</p>
</td>
Fyll inn 8
<tr>
Fyll inn 9
Ta kontakt på epost: <b><u>hjalmar.pjalle@mysilgoerra.uff</u></b>
</td>
</tr>
Fyll inn 10
</body>
</html>
```

TDT4105 SVARSKJEMA TIL OPPGAVE 1

Kandidatnummer: _____ Dato: _____ Side: _____ / _____

Oppgave	a	b	c
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			

TDT4105 SVARSKJEMA TIL OPPGAVE 1

Kandidatnummer: _____ Dato: _____ Side: _____ / _____

Oppgave	a	b	c
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			

List of built-in functions

abs(x)
absolute value of x, $|x|$

acos(x)
inverse cosinus of x, $\cos^{-1}(x)$

acosd(x)
inverse cosinus of x with angle in degrees

acot(x)
inverse cotangent of x, $\cot^{-1}(x)$

asin(x)
inverse sinus of x, $\sin^{-1}(x)$

asind(x)
inverse sinus of x with angle in degrees

atan(x)
inverse tangent of x, $\tan^{-1}(x)$

atand(x)
inverse tangent of x with angle in degrees

atan2(x,y)
element-wise inverse tangent of x/y

bar(x,y), bar(y)
produce bar graph of two vectors x and y
use indices of y as x-axis if one argument

cd *dir*
change current working directory

ceil(x)
return smallest integer $\geq$ x

char(x)
convert integer(s) into character(s)

clear var ...
delete variable(s) from the symbol table

cos(x)
cosinus of x with argument in radians

cosd(x)
cosinus of x with argument in degrees

cot(x)
cotangent of x with argument in radians

cotd(x)
cotangent of x with argument in degrees

cputime()
return total CPU time spent so far

disp(x)
print the value of x with a newline

eps, eps(N,M)
machine precision as a number or a N×M-matrix

exp(x)
compute the exponential of x, e^x

eye(N), eye(N,M)
return $N \times N$ or $N \times M$ identity matrix

false
return logical 0 (false)

status = fclose(fid)
close file with file-id *fid*
return 0 on success, -1 on failure

fix(x)
round x to nearest integer towards zero

floor(x)
return largest integer $\leq$ x

fid *or* [fid msg] = fopen(name,mode)
open file with pathname *name*, mode character is 'r':read, 'w':write, 'a':append, 'r+':read and write
return file-id *fid* > 0 on success, -1 on error
optionally return error message *msg*

fprintf(fid,format,variable,...)
print variable(s) with specified formatting
%mf, %m.nf or %f:fixed-form float-point number
%me, %m.ne or %e:float-point with exponent
%md or %d:integer, %s:string, %%:%, \n:newline
m is field width, n is number of digits in fraction

val *or* [val count] = fread(fid,sz,'double')
read *sz* elements from file *fid* to *val*
optionally return number of elements read in *count*
sz is any of inf:as much as possible, N:n elements, [N M]:N×M matrix, [N inf]:M as large as possible

status = fseek(fid,offset,origin)
set file position to *offset* within open file *fid*
origin is 'bof', 'cof' or 'eof'
return 0 on success, -1 on failure

position = ftell(fid)
return file-pointer position of *fid*, -1 on failure

count = fwrite(fid,var,'double')
write variable *var* to file *fid*
return number of elements written in *count*

grid on *or* off
turns grid on or off on a 2D-plot

hold off *or* **on**
do or do not erase previos plot before plotting next

val = input(msg) ***or*** **input(msg, 's')**
output *msg*, then read keyboard input to *val*
last form reads input as a string (does not evaluate)

intmax
return largest (32-bit) integer available

intmin
return smallest (32-bit) integer available

A_{inv} = inv(A)
return inverse of matrix A, the matrix A^{-1}

isfinite(x)
return 1 if x is a finite number, 0 otherwise

length(A)
return largest dimension of matrix A

val = load('-ascii', name)
load contents of text file *name* into *val*

log(x)
compute the natural logarithm, $\ln x$

log2(x)
compute the base-2 logarithm, $\log_2 x$

var *or* **[var ix] = max(x)**
find largest element in x, optionally with index *ix*

mesh(X,Y,Z)
plot 3-D mesh grid Z = f(X,Y)
use meshgrid to compute arrays X and Y

[X Y] = meshgrid(x,y) *or* **meshgrid(x)**
transforms domain specified by vectors (x,y) into arrays X and Y for use with 3-D plots
meshgrid(x) equals meshgrid(x,x)

var *or* **[var ix] = min(x)**
find smallest element in x, optionally with index *ix*

mod(x,y)
remainder of x/y, $x - \lfloor x/y \rfloor \cdot y$

nargin
number of arguments passed to the function

nargout
number of values the caller expects to receive

norm(x)
compute the 2-norm of x, `sqrt(sum(x.^2))`

str = num2str(x) ***or*** **num2str(x,n)**
convert input into text and store in *str*,
last form use a maximum precision of n digits

ones(N), ones(N,M)
return N×N or N×M matrix of ones

pause(secs), pause
pause execution *secs* seconds or until any key hit

pi, pi(N,M)
π as a number or a N×M-matrix

plot(x,y)
2-D plot of vector x versus vector y

prod(x)
product of elements in x, $\prod x_i$

pwd, string = pwd
print or return working directory as a string

rand, rand(N), rand(N,M)
return a random number on the open interval (0,1),
a N×N or N×M matrix of random numbers

realmax
return largest real (floating-point) number

realmin
return smallest real (floating-point) number

round(x)
round x towards nearest integer

save('-ascii', name, 'var', ...)
save variables *var, ...* on text file *name*

sign(x)
sign of x, -1 if negative, 0 if zero and 1 if positive

sin(x)
sine of x with argument in radians

sind(x)
sine of x with argument in degrees

size(A), size(A,n)
return all dimensions or n^{th} dimension of A

sort(X), sort(X,n), sort(X,n,mode)
sort X in ascending order, sort along n^{th} dimension
or sort with mode `'ascend'` or `'descend'`

sqrt(x)
compute square root of x, $\sqrt{x}$

x = str2double(string)
convert character string to (floating-point) number

A = str2num(string)
convert character string matrix to number
use str2double to convert a single number

y = sum(x)
compute sum of elements, $\sum x_i$

surf(X,Y,Z)
plot 3-D surface Z = f(X,Y)
use meshgrid to compute arrays X and Y

tan(x)
tangent of x with argument in radians

tand(x)
tangent of x with argument in degrees

tic, toc
set and check a wall-clock timer

true
return logical 1 (true)

type name
return the function or built-in matching *name*

version
return Matlab interpreter version string

who, who var, . . .
display all or specified variables *var, . . .*

whos, whos var, . . .
long form of who; more detailed listing

xlabel(str)
print x-axis label *str* onto 2D plot

ylabel(str)
print y-axis label *str* onto 2D plot

zeros(N)
return N×N or N×M matrix of zeroes