NTNU
Kunnskap for en bedre verden

Institutt for datateknikk og informasjonsvitenskap

Eksamensoppgave i TDT4105 Informasjonsteknologi - grunnkurs

Faglig kontakt under eksamen:	Rune Sætre	Mobil: 452 18103
	Anders Christensen	Mobil: 918 97181
	Nils Inge Rugsveen	Mobil: 958 76417

Eksamensdato:	**2014-12-06**
Eksamenstid (fra-til):	**09:00 – 13:00**
Hjelpemiddelkode/Tillatte hjelpemidler:	**Godkjent kalkulator**

Annen informasjon:

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 løse oppgavene. 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.

Målform/språk:	**Bokmål**
Antall sider:	**17 (inkl. Forside, svarark og appendiks)**

Innhold:

- Oppgave 1: Flervalgsoppgave (25%)
- Oppgave 2: Programmering: Fallskjerm (25%)
- Oppgave 3: Programmering: Værstasjon (30%)
- Oppgave 4: Kodeforståelse (20%)
- Appendiks: Nyttige funksjoner
- Svarark til Flervalgsoppgave (2 eksemplarer)

Kontrollert av:

Dato Sign

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. Hvilken fundamental aktivitet innen programvareutviklingsprosessen fokuserer på å endre programvaren for å møte endrede kunde- og markedskrav?
    a. Programvarespesifikasjon.
    b. Programvareutvikling.
    c. Programvarevalidering.
    d. Programvareevolusjon.
2. Hva er et analogt signal?
    a. Et kontinuerlig signal hvor den variable egenskap er gitt av en diskret funksjon, som gir verdier fra et definert og begrenset område.
    b. Et kontinuerlig signal hvor en variabel egenskap (f.eks. amplitude eller frekvens) representerer informasjonen som overføres.
    c. Et diskret signal som representeres ved hjelp av nuller og enere.
    d. En kombinasjon av alternativ a og b.
3. Hvilken type løkkestruktur er garantert å utføre handlingen minst en gang?
    a. pre-test løkke (pretest loop).
    b. post-test løkke (posttest loop).
    c. begge typer.
    d. ingen av typene.
4. Omtrent hvor mange ganger raskere er en 1 GHz - prosessor i forhold til en på 2 MHz.
    a. Halvparten så rask.
    b. Like rask.
    c. Dobbelt så rask.
    d. 500 ganger så rask.
5. Hva sier Nyquist-regelen?
    a. at samplingsrate ved lyd må være minst det dobbelte i forhold til høyeste frekvensen.
    b. at lyd over 20000Hz ikke kan høres av det menneskelige øret.
    c. at tapsfri komprimering ikke er mulig for lyd.
    d. at lyddata tapsfritt kan komprimeres med maksimalt en faktor 2*pi.
6. Hvilken programvareprosessmodell bør velges for et prosjekt der det skal utvikles et helt nytt system hvor eksisterende komponenter ikke finnes og kunden er usikker på hvordan systemet skal være?
    a. Vannfallsmodellen.
    b. Inkrementell utvikling.
    c. Gjenbruksorientert systemutvikling.
    d. Havmodellen.
7. Hva er hensikten med et paritetsbit i digitale signaler?
    a. Forteller hvor meldingen skal sendes.
    b. Gjør meldingene raskere å overføre (komprimering).
    c. Bidrar til å detektere feil i digitale signaler.
    d. Krypterer signaler så overføringen av data blir sikrere.

8. Kompleksiteten til sortering ved innsetting (insertion sort) er
    a. $\Theta(n)$.
    b. $\Theta(n \log n)$.
    c. $\Theta(n^2)$.
    d. $\Theta(2n)$.

9. En moderne prosessor er typisk bygd opp av mange millioner små…
    a. Dioder.
    b. Magneter.
    c. Transistorer.
    d. Kondensatorer.

10. En byte med minne i datamaskinen kan lagre hvor mye?
    a. 16 bits.
    b. 8 flyttall.
    c. fire ASCII-tegn.
    d. en heltallsverdi mellom 0 og 255.

11. Hvilken av de følgende er en kjent fordel med vannfallsmodellen?
    a. Tar hensyn til brukerkrav som endrer seg i løpet av prosjektet.
    b. Gjør prosessen synlig og enklere å monitorere for prosjektlederen.
    c. Får tidlige versjoner av systemet raskt ut til kunden.
    d. Åpner for kontinuerlig tilbakemelding fra brukerne av systemet.

12. Morsekode representerer bokstaver som sekvenser av prikk og strek som er
    a. like lange for alle bokstaver i alfabetet.
    b. kortere for bokstaver tidlig i alfabetet, lenger for bokstaver sist i alfabetet.
    c. kortere for vokaler, lenger for konsonanter.
    d. kortere for bokstaver som forekommer hyppig i vanlig tekst, lenger for sjeldnere bokstaver.

13. Hvilken av disse er en korrekt gjengivelse av teoribokas definisjon av en algoritme? "En algoritme er et ordnet sett av…"
    a. "… entydige, utførbare skritt som definerer en terminerende prosess" (unambiguous, executable steps that defines a terminating process).
    b. "… entydige, effektive skritt som definerer en utførbar prosess" (unambiguous, efficient steps that defines an executable process).
    c. "… velformede, effektive skritt som definerer en terminerende prosess" (well-formed, efficient steps that defines a terminating process).
    d. "…velformede, utførbare skritt som definerer en effektiv prosess" (well-formed, efficient steps that defines an efficient process).

14. En datamaskin går i en uendelig løkke som kalles
    a. Det naturlige kretsløpet.
    b. Hent-Utfør kretsløpet.
    c. Det evige kretsløpet.
    d. Beregnings-kretsløpet.

15. Hva er korrekt binær representasjon av 'NTNU' i 8 bits ASCII?
    a. `01001110 01010100 01001110 01010101.`
    b. `01100001 01100100 01110011 01100110.`
    c. `01101110 01110101 01110100 01001110.`
    d. `01100010 01010101 01010010 01010000.`

16. I hvilket tilfelle er det mest nyttig å bruke gjenbruksorientert systemutvikling?
    a. Når det finnes tilgjengelig programvare som kan gjøre oppgaver systemet skal utføre.
    b. Når det skal lages programvare for å håndtere resirkulering av søppel eller lignende systemer.
    c. Når det skal gjenbrukes ideer fra tidligere prosjekter.
    d. Når det skal gjenbrukes systemutviklere og systemdesignere fra tidligere prosjekter.

17. Hva står forkortelsen ISP for?
    a. Internet Service Provider.
    b. Information Security Protocol.
    c. Internet Security Protocol.
    d. Information Super Pool.

18. Kompleksiteten til binærsøk er
    a. $\Theta(n)$ hvis lista er sortert og $\Theta(n \log n)$ hvis den er usortert.
    b. $\Theta(\log n)$ hvis lista er sortert og $\Theta(2 \log n)$ hvis lista er usortert.
    c. $\Theta(\log n)$ hvis lista er sortert og $\Theta(n)$ hvis lista er usortert.
    d. $\Theta(\log n)$ hvis lista er sortert. Binærsøk er ubrukelig hvis lista er usortert.

19. RAM
    a. Husker alle verdiene når strømmen kuttes.
    b. Er alltid inndelt i blokker på 1 kilobyte.
    c. Betyr Random Access Memory.
    d. Kan trygt fjernes uten at maskinen slutter å fungere.

20. Hva står bokstavene i RGB for?
    a. Red, Green, Blue.
    b. Readable Graphics Byte.
    c. Raster Grayscale Balance.
    d. Realtime GPU Backlog.

21. Hva kalles aktiviteten som har fokus på å identifisere den overordnede strukturen for et system inkludert dets sub-systemer?
    a. Hoveddesign.
    b. Arkitekturdesign.
    c. Interfacedesign.
    d. Komponentdesign.

22. MODEM er en forkortelse for
    a. MOdulator / DEModulator.
    b. Massive Online Digital Electric Messaging.
    c. MOnitored Data EMission.
    d. Mapping Of Digital Electronic Mail.

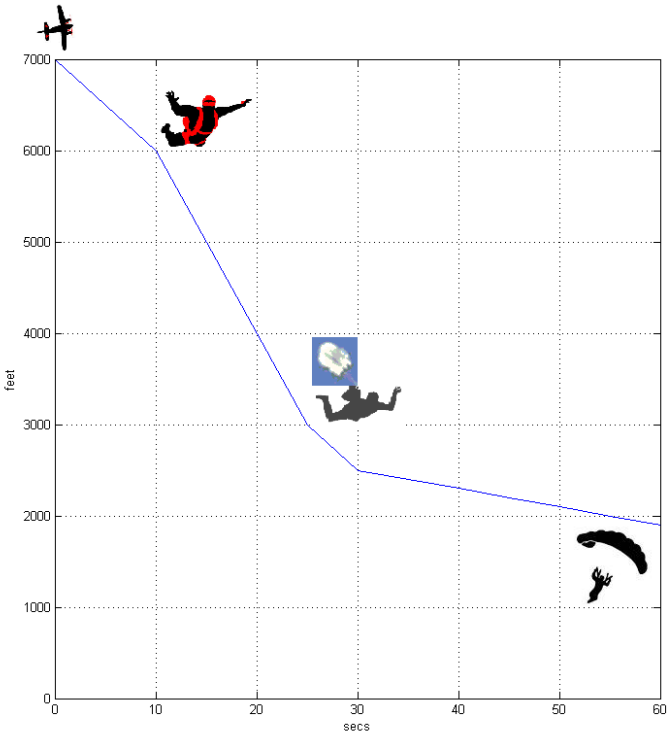
23. ASCII-kode representerer bokstavene A til Z som sekvenser av 0 og 1 som er
    a. like lange for disse bokstavene i alfabetet.
    b. kortere for bokstaver tidlig i alfabetet, lenger for bokstaver sist i alfabetet.
    c. kortere for vokaler, lenger for konsonanter.
    d. kortere for bokstaver som forekommer hyppig i vanlig tekst, lenger for sjeldnere bokstaver.

24. Et nettverk som knytter sammen datamaskiner og utstyr i et begrenset område som et kontor, bygning eller i en bolig betegnes med forkortelsen:
    a. LAN.
    b. MAN.
    c. PAN.
    d. WAN.
25. VPN (Virtual Private Network) kan gi mottageren inntrykk av at en reisende ansatt sin bærbare PC befinner seg innenfor bedriftens nettverk ved at meldinger fra denne PC'en
    a. plasseres inni en kryptert datapakke for ekstern oversendelse.
    b. sendes med en tidsforsinkelse.
    c. sendes ekstra hurtig, med høy prioritet.
    d. sendes med en falsk avsenderadresse som inneholder et virus.

RIKTIGE LØSNINGER:
1: D, 2: B, 3: B, 4: D, 5: A
6: B, 7: C, 8: C, 9: C, 10: D
11: B, 12: D, 13: A, 14: B, 15: A
16: A, 17: A, 18: D, 19: C, 20: A
21: B, 22: A, 23: A, 24: A, 25: A

Oppgave 2 Programmering Fallskjerm (25%)

(I denne oppgaven kan det være gunstig å kalle funksjoner som du har laget i tidligere deloppgaver. Selv om du ikke har fått til den tidligere oppgaven, kan du kalle funksjon derfra med antagelse om at den virker som spesifisert i oppgaveteksten.)
NTNU fallskjermklubb (NTNU-FSK) trenger hjelp til å lage et nytt opplærings- og administrasjonsprogram. De har bedt firmaet ditt (ITGK) om hjelp, og du har fått jobben med å programmere funksjoner som beskrevet i deloppgavene under.
Vi benytter en litt forenklet versjon av jordens fysiske lover:



Figur 1. Hopp fra 7000 fot.

En fallskjermhopper faller (med konstant/gjennomsnittlig hastighet) 100 fot pr. sekund, de 10 første sekundene, og deretter med konstant topphastighet på 200 fot pr. sekund til skjermen må åpnes i 3000 fots høyde (se figur 1). Hvis man mot normalt hopper ut under 3000 fot må skjermen utløses umiddelbart (etter 0 sekunder).

Medlemsdatabasen til NTNU-FSK ligger lagret på en fil 'members.txt', med følgende format:

NAVN;IDNR;VEKT;SKJERMST.
Eksempel på innholdet i filen:

```
Frank Stank;D-49334;75;120
Bjarne Stor;C-49335;95;150
Dumbo Ear;D-50105;450;750
Peter Pan;A-12345;30;100
```

Oppgave 2a (5%)
Lag en funksjon `inputPerson` som leser inn navn, id, vekt og skjermstørrelse fra tastaturet, og returnerer en struct `person` med riktige verdier i følgende felt: name, id, weight, size. Merk at «name» og «id» er tekststrenger, mens «weight» og «size» er tall. Du kan anta at bruker skriver inn lovlige verdier.

Eksempel på kjøring (tekst med understrekning skrives inn av brukeren):

```
>> person = inputPerson( )
Name: Fredrik Olsen
ID: B-77777
Kg: 80
Size: 240
person =
       name: 'Fredrik Olsen'
         id: 'B-77777'
     weight: 80
       size: 240
>>
```

Mulig løsning 2a:

```
function person = inputPerson()
      person.name = input( 'Navn: ', 's' );
      person.id = input( 'ID: ', 's' );
      person.weight = input( 'Kg: ' );
      person.size = input( 'Size: ' );
end
```

Oppgave 2b (5%)
Lag en funksjon `readDbFile` som leser inn hele medlemsbasen i en struct-vektor, med følgende feltnavn: «name», «id», «weight» og «size». «name» og «id» skal lagres som tekst, mens «weight» og «size» skal lagres som tall. Du kan anta at filen finnes og at det ikke oppstår noen problemer ved åpning/lukking, og at filen ikke inneholder noen blanke eller ugyldige linjer. Funksjonen skal ha inn-parameter `filename` og retur-verdi `db`.
Eksempel på kjøring:

```
>> db = readDbFile( 'members.txt' )
db = 1x4 struct array with fields:
    name
    id
    weight
    size
>> db(1)
ans =
      name: 'Frank Stank'
        id: 'D-49334'
    weight: 75
      size: 120
>>
```

Mulig løsning 2b:

```
function db = readDbFile( filename )
indeks = 1;
fid = fopen( filename, 'r');
while ~feof( fid )
    line = fgetl( fid );
    [name, rest] = strtok( line, ';' );
    [id, rest] = strtok( rest, ';' );
    [weight, rest] = strtok( rest, ';' );
    [size, ~] = strtok( rest, ';' );

    person = struct( 'name',name,'id',id,'weight', str2double( weight ),...
        'size', str2double( size ) );
    db( indeks ) = person;
    indeks = indeks +1;
end %while more lines
fclose( fid );
end %function
```

```
%Løsning med strsplit
function db = readDbFile( filename )
indeks = 1;
fid = fopen( filename ); %, 'r'); % Deafault 'read'
while ~feof( fid )
    line = fgetl( fid );
    p = strsplit( line, ';' );
    person = struct( 'name',p{1},'id',p{2},'weight', str2double( p{3} ),...
        'size', str2double( p{4} ) );
    db( indeks ) = person;
    indeks = indeks +1;
end %while more lines
fclose( fid );
end %function
```

Oppgave 2c (5%)
Lag en funksjon `printMemberList` som skriver ut følgende overskrifter og innholdet i struct-vektoren (`db` fra ovenfor) på skjermen på følgende format:

NAVN (avsatt 15 tegn) ID-NR (avsatt 9 tegn) VEKT (avsatt 5 siffer/tegn) kg. FALLSKJERMST (avsatt 4 siffer/tegn) kvadratfot

Du kan anta at databasen ikke har innhold som går utover de avsatte antall tegn for hver felt. Feltene skal være høyrejusterte.

Inn-parameter `db`, ingen retur-verdi.
Eksempel på bruk:

```
>> db = readDbFile('members.txt');
>> printMemberList( db )
           NAVN    ID-NR VEKT kg. SKJERMSTØRRELSE
    Frank Stank  D-49334   75 kg.  120 kvadratfot
    Bjarne Stor  C-49335   95 kg.  150 kvadratfot
      Dumbo Ear  D-50105  450 kg.  750 kvadratfot
      Peter Pan  A-12345   30 kg.  100 kvadratfot
>>
```

Mulig løsning 2c:

```
function printMemberList( db )
      fprintf( '%15s%9s%5s kg. %s\n','NAVN','ID-NR','VEKT','SKJERMSTØRRELSE');
      for i=1:length(db)
          fprintf( '%15s%9s%5d kg. %4d kvadratfot\n', db(i).name, ...
              db(i).id, db(i).weight, db(i).size );
      end %for
end
```

Oppgave 2d (5%)
Lag en funksjon `addPerson` med inn-parametere `filename` (der databasen er lagret). Funksjonen skal be bruker skrive inn informasjon om en person beskrevet med navn, id, vekt og skjermstørrelse, og lagre dette i variabelen `person`. Funksjonen skal så lese inn databasen som ligger lagret i filen `filename` datastrukturen (struct-vektor `db`). Deretter skal opplysningene om den nye person legges til i `db` og i filen `filename`. Bruk riktig format: se «Eksempel på innholdet i filen» ved figur 1 (se side 6). Returverdi `db` skal inneholde den oppdaterte databasen.

Du trenger ikke å skrive kode for feilhåndtering i forbindelse med å lese eller skrive fra/til fil.
Eksempel på kjøring (Endringen er markert med **fet skrift**):

```
>> db = addPerson( 'members.txt' )
Name: Santa Klaus
ID: H-12345
Kg: 155
Size: 380
db =
1x5 struct array with fields:
    name
    id
    weight
    size
>> db(5)
ans =
     name: 'Santa Klaus'
       id: 'H-12345'
   weight: 155
     size: 380
>>
```

Mulig løsning 2d:

```
function db = addPerson( filename )
      % Add the new person in the end of the db.
      % Write the new db to the given filename, and return the new struct-vector
      db = readDbFile( filename );
      person = inputPerson();
      db = [db person];

      fid = fopen( filename, 'w' );
      for i=1:length( db )
          fprintf( fid, '%s;%s;%d;%d\n', db(i).name, db(i).id, ...
              db(i).weight, db(i).size );
      end %for
      fclose( fid );
end

%% Mer elegant løsning

function db = addPerson( filename )
      %% addPerson  Add new Skydiver to the database.
      p = inputPerson();
      fid = fopen( filename, 'a' );
      fprintf( fid, '%s;%s;%d;%d\n', p.name, p.id, p.weight, p.size );
      fclose( fid );
      db = readDbFile( filename );
end
```

Oppgave 2e (5%)

For en fallskjermhopper er det veldig viktig å være klar over hvor mange sekunder man kan vente før man må åpne fallskjermen (Se figur 1). Lag en funksjon `feet2seconds` som regner ut hvor mange sekunder det tar å falle fra et oppgitt antall fot ned til 3000 fot (inn-parameter `feet`, og retur-verdi `seconds`). Bruk informasjon gitt i starten av oppgave 2 (forklaringen til figur 1) til å beregne riktig tid. Hvis antall fot er under 3000 skal funksjonen returnere 0.
Eksempler på bruk:

```
>> feet2seconds( 12500 )
ans =   52.5000
>> feet2seconds( 7000 )
ans =    25
>> feet2seconds( 2000 )
ans =     0
```

Mulig løsning 2e:

```
function seconds = feet2seconds( feet )
      if feet > 4000
          seconds = 10 + (feet-4000) / 200;
      elseif feet > 3000
          seconds = (feet-3000)/100;
      else
          seconds = 0;
      end % if
end
```

Oppgave 3 Programmering Værstasjon (30%)

Du skal behandle data fra en værstasjon for et større antall dager. Dataene ligger lagret som tall i en tabell som heter `weatherData`. Denne tabellen har tre kolonner, én for hver type måledata, og den har en rad for hver dag. De tre typene med måledata er maksimumstemperatur, minimumstemperatur og nedbørsmengde. Vi refererer til dagene slik at dataene på første rad er for dag nr 1, dataene på andre rad er for dag nr 2 osv. Eksempel på utskrift av `weatherData` kan være:

```
>> weatherData
ans =
   12.0000    2.4000    8.2000
    6.1000    0.6000   11.9000
    8.3000   -3.5000    0.0000
   11.6000   -5.2000    0.0000
   15.3000    2.8000   14.3000
>>
```

Oppgave 3a (10%)

Skriv en funksjon som du kaller `weatherStats` og som tar variabelen `weatherData` som parameter. Funksjonen skal gå igjennom dataene og utfra det skrive et sammendrag som vist under. Det vil si at den skal skrive ut antall dager i perioden, totalmengden av nedbør i hele perioden, samt at den skal liste den aller laveste og aller høyeste temperaturen sammen med nummeret på dagene for disse temperaturene.

Eksempel på kjøring for dataene vist i den grå boksen ovenfor:

```
>> weatherStats(weatherData)
There are 5 days in the period.
The highest temperature was 15.3C, on day number 5.
The lowest temperature was -5.2C, on day number 4.
There was a total of 34.4mm rain in the period.
>>
```

Mulig løsning 3a:

```
%% Antar at brukeren er interessert i den første høyeste/laveste i tilfelle flere
like.

Løsningsforslag med løkker:

function weatherStats( weatherData )
   days = size(weatherData,1) ;
   fprintf('There are %d days in the period.\n', days );

   maxtemp = weatherData(1,1) ;
   maxday = 1 ;
   mintemp = weatherData(1,2) ;
   minday = 1 ;
   sum = weatherData(1,3) ;

   for n=2:days
      if weatherData(n,1) > maxtemp
         maxtemp = weatherData(n,1) ;
         maxday = n ;
      end %if

      if weatherData(n,2) < mintemp
         mintemp = weatherData(n,2) ;
         minday = n ;
      end %if

      sum = sum + weatherData(n,3) ;
   end %for

   fprintf('The highest temperature was %.1fC, on day number %d.\n', ...
          maxtemp, maxday ) ;
   fprintf('The lowest temperature was %.1fC, on day number %d.\n', ...
          mintemp, minday ) ;

   fprintf('There was a total of %.0fmm rain in the period.\n', ...
          sum);

   end %function

Løsning med Matlab-funksjoner

function weatherStats( weatherData )
   days = size(weatherData,1) ;
   fprintf('There are %d days in the period.\n', days );

   day = find(  weatherData(:,1) == max(weatherData(:,1)) , 1 ) ;
   fprintf('The highest temperature was %.1fC, on day number %d.\n',  ...
      weatherData(day,1), day);

   day = find(  weatherData(:,2) == min(weatherData(:,2)) , 1 ) ;
   fprintf('The lowest temperature was %.1fC, on day number %d.\n', ...
       weatherData(day,2), day ) ;

   fprintf('There was a total of %.0fmm rain in the period.\n', ...
       sum(weatherData(:,3)));
end %function
```

Oppgave 3b (10%)
Skriv en funksjon `coldestThreeDays` som tar som parameter variabelen `weatherData`. Funksjonen skal finne den perioden av tre sammenhengende dager som hadde den laveste gjennomsnittlige minimumstemperaturen. Den skal returnere nummeret på første dagen i denne tredagersperioden. Dersom det er flere perioder som er like kalde, så skal den returnere kun den siste av disse periodene. Et eksempel på en kjøring av denne funksjonen for `weatherData` som definert tidligere i oppgaven gir:

```
>> coldestThreeDays(weatherData)
ans =
    2
>>
```

Mulig løsning 3b:

```
%% Vi antar at det er minst tre dager i perioden.

function start = coldestThreeDays( weatherData )
      days = size(weatherData,1) ;
      lowest = weatherData(1,2) + weatherData(2,2) + weatherData(3,2) ;
      start = 1 ;       %ALTERNATIV: test = lowest ;

      for n=2:days-2
            test = weatherData(n,2) + weatherData(n+1,2) + weatherData(n+2,2) ;
%ALTERNATIV: test = test - weatherData(n-1,2) + weatherData(n+2,2)

            if (test <= lowest)
                  lowest = test ;
                  start = n ;
            end %if
      end %for
end %function
```

Oppgave 3c (10%)
Værstasjonen har nettopp rapportert data for ytterligere en dag. Den kommer som en tekststreng lagret i variabelen `extraData` som har formatet som vist under:

```
>> extraData = 'max=23.5, min=9.3, 5.1mm';
>>
```

Skriv en funksjon `addNewDay` som tar `extraData` samt `weatherData` som parametere, og som returnerer en ny versjon av `weatherData` som er oppdatert med de nye dataene på slutten av tabellen. Under er vist et eksempel på kjøring (endring er vist i **fet skrift**):

```
>> updated = addNewDay( extraData, weatherData ) ;
>> updated(end-2:end,:)
ans =
   11.6000   -5.2000    0.0000
   15.3000    2.8000   14.3000
   23.5000    9.3000    5.1000
>>
```

Mulig Løsning 3c:

```
Løsning med bruk av sscanf()

function weatherData = addNewDay( extraData, weatherData )
  [ newData ] = sscanf(extraData, 'max=%f, min=%f, %fmm') ;
  weatherData(end+1,:) = ...
             [ newData(1), newData(2), newData(3) ];
end %function

Løsning med bruk av strtok()

function weatherData = addNewDay( extraData, weatherData )
   [ junk, rest ] = strtok(extraData,'=') ;
   [ maxtemp, rest ] = strtok(rest(2:end), ',') ;
   [ junk, rest ] = strtok(rest(2:end),'=') ;
   [ mintemp, rest ] = strtok(rest(2:end), ',') ;
   [ rain, rest ] = strtok(rest(2:end), 'm') ;
   weatherData(end+1,:) = [ str2num(maxtemp), ...
       str2num(mintemp), str2num(rain) ] ;
end %function

Løsningsforslag med strsplit() og ismember()

function weatherData = addNewDay(extraData, weatherData)
   extraData(find(ismember(extraData, 'minax,='))) = [] ;
   data = strsplit(extraData) ;
   weatherData(end+1,:) = [ str2num(data{1}), ...
          str2num(data{2}), str2num(data{3}) ] ;
end %function
```

```
Løsningsforslag med indeksering av strenger

function weatherData = addNewDay(extraData, weatherData)
   i=5 ;
   while i<length(extraData) && extraData(i)~=','
     i = i + 1 ;
   end %while
   newMax = str2num(extraData(5:i-1)) ;

    i = i + 6 ;
    start = i ;
    while i<length(extraData) && extraData(i)~=','
      i = i + 1 ;
    end %while
    newMin = str2num(extraData(start:i-1)) ;

    i = i + 2 ;
    start = i ;
    while i<length(extraData) && extraData(i)~='m'
      i = i + 1 ;
    end %while
    newRain = str2num(extraData(start:i-1)) ;

    weatherData(end+1,:) = [ newMax newMin newRain ] ;
end %function
```

Oppgave 4 Kodeforståelse (20%)

Oppgave 4a (5%)
Hva returneres ved kjøring av funksjonen `myst([1,2,3,3,2,1])` med kode som vist under? (3 %)
Forklar med en setning hva funksjonen `myst` gjør? (2 %)

```
function out = myst(A)
  out = 0;
  L = length(A);
  if ( L > 1 )
    B = A(1) * A(L);
    out = B + myst( A(2:L-1) );
  end %if
end %func
```

Løsning 4a:

ans =
   14
Forklaring: Går fra ytterst mot midten og multipliserer to og to ledd, og legger sammen resultatet (1*1+2*2+3*3).

Oppgave 4b (5%)
Hva blir skrevet ut når man kjører funksjonen `myst_b()` nedenfor? (3 %)
Forklar med en setning hva funksjonen `mystery` gjør? (2 %)

```
function myst_b( )
  disp( mystery(4) )
end
```

```
function out = mystery( W )
  %Creating a Matrix with W x W zeros
  table = zeros( W );
  for a = 1:W
    table(a,a)=1;
    b=0;
    while a-b > 1 && a+b < W
      b = b+1;
      table( a-b, a+b ) = 1;
      table( a+b, a-b ) = 1;
    end
  end
  out = table;
end
```

Løsning 4b:

```
     1     0     1     0
     0     1     0     1
     1     0     1     0
     0     1     0     1
```

Forklaring: Lager et diagonalt mønster av 1'ere for annen hver rekke/kolonne i en W x W tabell av 0'ere («Sjakk-brett»).

Oppgave 4c (5%)
Hva returneres ved kjøring av funksjonen `myst_c('RBHOOASDUEØGNGEBLSOIURMNGTD')` med kode som vist under? (3 %)
Forklar med en setning hva funksjonen `myst_c` gjør? (2 %)

```
function B = myst_c(A)
  B='';
  for x = 1:3:length(A)
    B = [B A(x)];
  end %for
end %func
```

Løsning 4c:

```
ans =
ROSENBORG
```

Forklaring: Returnerer en streng med hvert tredje tegn, fra og med det første, fra parameteren A.

Oppgave 4d (5%)
I et program som skal trene ungdomsskoleelever i matematikk, trenger vi en funksjon for å sjekke korrekt nøsting av parenteser i uttrykk der tre ulike parentestyper er tillatt. Funksjonen trenger ikke å sjekke at uttrykket ellers er fornuftig, kun at parenteser kommer i lovlig rekkefølge og går opp mtp antall og plassering av alle start- og sluttparenteser. Vi har også skrevet tre setninger som kaller funksjonen for å teste om den virker. Koden er vist her (linjenummer til venstre ikke del av koden men er tatt med så du lettere kan vise til spesifikke kodelinjer i svaret ditt):

```
1 function check_main( )
2     A = check( '{a+4*[b-2*(c+5)]/11}' ) %  should be  true
3     B = check( '{a+4*[b-2*(c+5])/11}' ) %  should be  false
4     C = check( '{a+4*[b-2*(c+5])/11}}' ) % should be  false
5 end %function check_main

6 function out = check( expression )
7     PAREN = '([{)]}';
8     parenthesis_list = [];
9     for char = expression
10        if find( char == PAREN(1:3) ) %start-parenthesis found
11            %add corresponding end-parenthesis to the end of the list!
12            parenthesis_list =  [ parenthesis_list...
13                                  PAREN( find( PAREN == char )+3 ) ];
14        elseif find( char == PAREN(4:6) ) % end-parenthesis found
15            if find( char == parenthesis_list ) % matched start/end-parenthesis
16                parenthesis_list( char == parenthesis_list ) = []; %remove it
17            else
18                out = false;
19                return
20            end
21        end
22    end
23    out = parenthesis_list;
24 end
```

Som kommentarene sier skulle utskriften fra de tre setningene ha blitt hhv true (1), false (0) og false (0). Når vi kjører `check_main`-funksjonen får vi imidlertid:

```
>> check_main()
A =   Empty string: 1-by-0
B =   Empty string: 1-by-0
C =     0
```

Det er to feil i koden:

1) funksjonen returnerer `Empty string: 1-by-0` for A: og B: i stedet for boolske variable. Hvis denne feilen blir rettet, vil kjøring av programmet indikere den andre feilen:

```
>> check_main()
A =     1
B =     1
C =     0
```

2) som man ser over: funksjonen returnerer true (1) for noen uttrykk som skulle gitt false (0), som for linje B over

Spørsmål: Forklar hvilke kodelinjer som forårsaker feil 1) og feil 2) og hvordan de enklest kan rettes. I begge tilfeller skal det være mulig å rette feilen bare ved å endre noe i eksisterende kodelinjer, det skal ikke være nødvendig å legge til nye kodelinjer.

Løsning 4d:

1) Må endre linje 23

```
23    out = isempty( parenthesis_list );
```

2) Må endre linje 15 og 16:

```
15          if char == parenthesis_list ( end ) % matched end-parenthesis
16              parenthesis_list( end ) = []; %remove it
```

Forklaring:
Linje 23 returnerte det som var igjen av liste. Endret til å sjekke om lista var tom!

Linje 15 sjekket ikke rekkefølge. Endret til å bare sjekke med parentes som ble sist lagt til lista.
Linje 16 fjernet alle forekomster av gitt end-parentes. Endret til å fjerne den som ble sist lagt til lista.

*Svarskjema flervalgsoppgave (sjablong – **korrekte svar synlig**)*

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				
1.21				
1.22				
1.23				
1.24				
1.25				

*Svarskjema flervalgsoppgave (sjablong – **feil svar synlig**)*

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				
1.21				
1.22				
1.23				
1.24				
1.25				