NTNU

Kunnskap for en bedre verden

TDT4105 Informasjonsteknologi, grunnkurs

Matlab: Søking

Amanuensis Terje Rydland
Kontor: ITV-021 i IT-bygget vest (Gløshaugen)
Epost: terjery@idi.ntnu.no
Tlf: 735 91845

Kunnskap for en bedre verden

Søkealgoritmer

- En søkealgoritme er en algoritme som søker gjennom en datamengde, på jakt etter en bestemt verdi.
- Vi skal begrense oss til det enkleste, søking i lister
- Vi skal se på to typer søk:
  - Sekvensielt søk
  - Binærsøk

3

Sekvensielt søk

- Går gjennom alle dataelementene fra begynnelse til slutt
- Tidkrevende, må gå gjennom alle elementene
- Fordringsløs – krever ikke noen forhåndsinformasjon om datamengden
- Eksempel:
  - Finne ut om et telefonnr finnes i telefonkatalogen

NTNU

4

finnesTall.m

```
function funnet = finnesTall(liste, tall)
% Returnerer true hvis tall finnes i liste, false ellers
% Sekvensiell søkning

    funnet = false;

    for i = 1:length(liste)
        if liste(i) == tall
            funnet = true;
            return
        end %if
    end %for

end % function
```

NTNU

5

Varianter

- Finne antall forekomster at tallet i listen
- Finne første indeks (posisjon) til tall i listen
- Finne alle indeksene til tall i listen

- Prøv gjerne å programmere disse selv!

- Sekvensielt søk er så langsomme at man søker alternativer:
  - Sortering og så søking
  - Søketrær (indekser)

NTNU

6

Binærsøking

- Hvis datamengden er sortert (ordnet) kan søket gjøres mye mer effektivt
- Sjekker midterste element i datamengden
  - Elementet er der -> ferdig
  - Fortsetter å lete i den relevante halvparten
- Arbeidsmengde
  - Antall halveringer ca $\log_2(N)$
  - Sekvensiell søking: ca N.
  - Stor forskjell når N er stor

NTNU

7

Koding av binærsøk

- Koder b_finnesTall(liste, tall) med binærsøking
- Forutsetter at liste er sortert stigende
- Bruker l(ow) og h(igh) for å avgrense et mindre og mindre område der tall kan finnes
- Regner ut elementet midt i dette området
  - median = floor((l+h)/2)
  - Husk at floor runder nedover til nærmeste heltall

NTNU

8

b_finnesTall(liste,tall)

```
function funnet = b_finnesTall(liste,tall)
% Returnerer true hvis tall finnes i liste, false ellers
% Bruke binærsøk, lista antas å være sortert stigende

    l = 1;              % Minste indeks
    h = length(liste);  % Største indeks

    while l <= h
        % Finner midtpunkt
        median = floor((l+h)/2);

        if liste(median) == tall
            l = h+1; % Tall er funnet. Dette avslutter løkken
        elseif liste(median) < tall
            % Må lete i øvre halvdel
            l = median + 1;
        else
            % Må lete i nedre halvdel
            h = median - 1;
        end %if
    end % while

    % Setter returverdi
    funnet = liste(median) == tall;

end % function
```

NTNU

b_finnesTall(minliste, 8) - programsporing

Steg	liste	tall	l	h	median	funnet
1	[1 8 16 23 30]	8	-	-	-	-
2			1	-	-	-
3				5	-	-
4					3	-
5				2		-
6					1	-
7			2			-
8					2	-
9			3			-
10						true

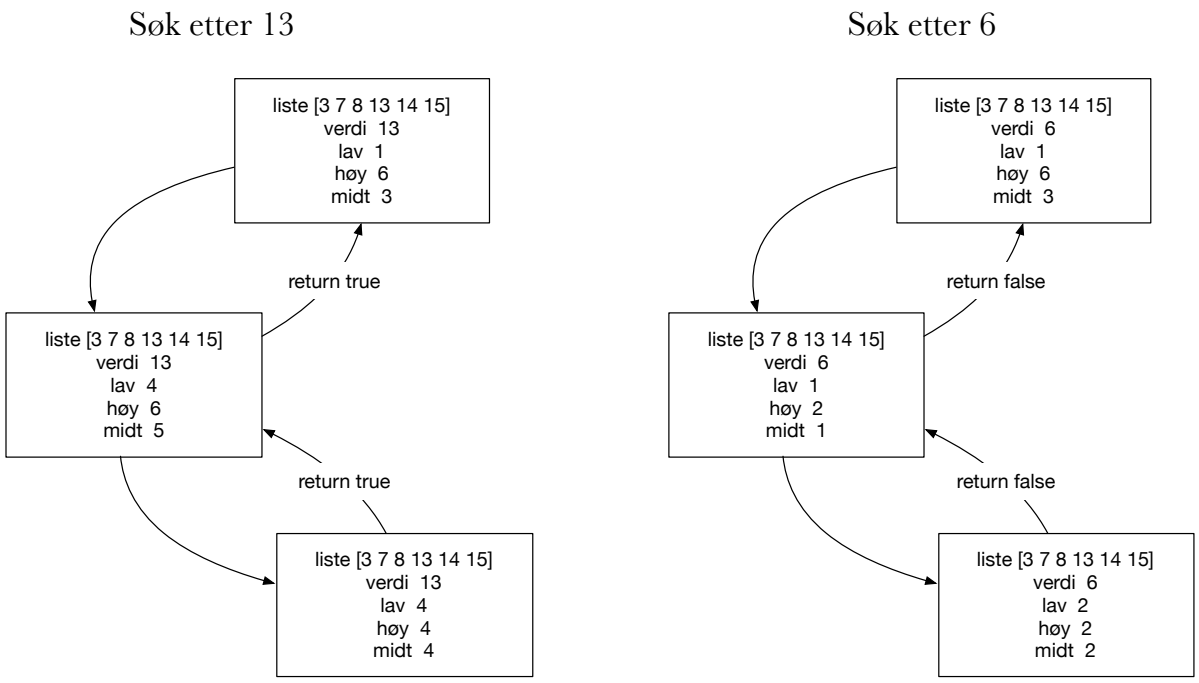
b_finnesTall(minliste, 10) - programsporing

Steg	liste	tall	l	h	median	funnet
1	[1 8 16 23 30]	10	-	-	-	-
2			1	-	-	-
3				5	-	-
4					3	-
5				2		-
6					1	-
7			2			-
8					2	-
9			3			-
10						false

Binærsøking rekursivt

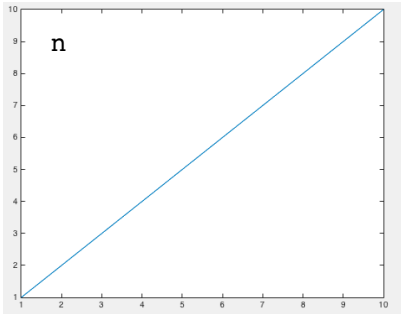
```
function funnet = binaersoek(liste, verdi, lavIndeks, hoyIndeks)
% Rekursivt binærsøk etter verdi i liste

    midten = floor((hoyIndeks + lavIndeks)/2);

    if lavIndeks > hoyIndeks
        funnet = false;
    elseif verdi == liste(midten)
        funnet = true;
    elseif verdi < liste(midten)
        funnet = binaersoek(liste, verdi, lavIndeks, midten - 1);
    else
        funnet = binaersoek(liste, verdi, midten + 1, hoyIndeks);
    end % if

end % function
```


Binærsøking rekursivt graf

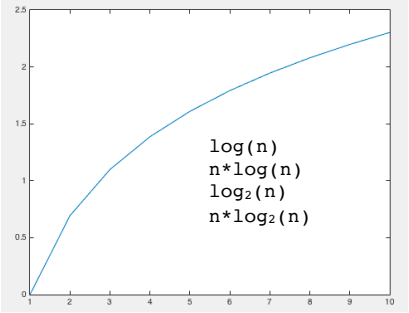


Kompleksitet for søking

- Sekvensiell søking vokser med N
  - Kjøretiden vil bli $k_1*N + k_0$
  - For stor N dominerer k_1*N
  - Vi sier at algoritmen er O(N)
  - Vokser N, øker kjøretiden proporsjonalt med N



- Binærsøking gjør $\log_2(N)$ sammenligninger
  - Kjøretiden vil bli $k_1*\log_2(N) + k_0 = k*\log(N) + k_0$
  - Vokser N, øker kjøretiden proporsjonalt med log(N)
  - Vi sier at algoritmen er O(log(N))



NTNU

Kunnskap for en bedre verden

TDT4105 Informasjonsteknologi, grunnkurs

Eksempler

Amanuensis Terje Rydland
Kontor: ITV-021 i IT-bygget vest (Gløshaugen)
Epost: terjery@idi.ntnu.no
Tlf: 735 91845

Kunnskap for en bedre verden

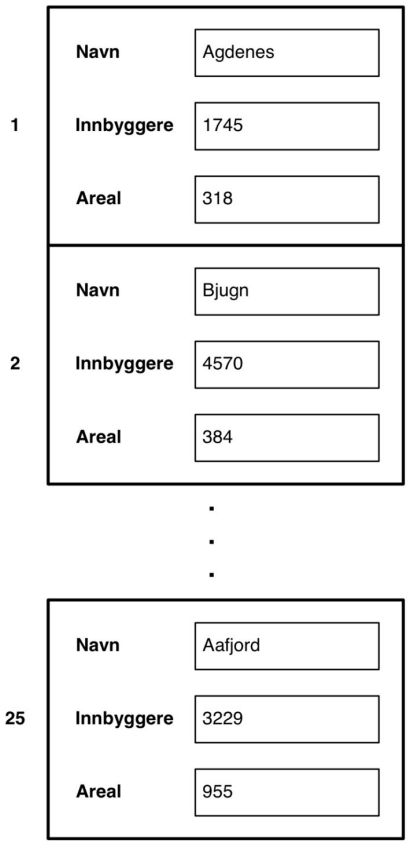
Kommunedata

- Sør-Trøndelag
  - 25 kommuner
    - Navn
    - Innbyggere
    - Areal (km^2)
  - Tekstfil med 25 linjer
- Må representere dette i programmet
  - Datastruktur
  - Mulige løsninger?

Agdenes	1745	318
Bjugn	4570	384
Frøya	4326	241
Hemne	4232	670
Hitra	4340	686
Holtålen	2048	1210
Klæbu	5894	186
Malvik	12677	169
Meldal	3903	613
Melhus	15114	695
Midtre-Gauldal	6050	1861
Oppdal	66912	274
Orkdal	11365	594
Osen	1025	387
Rennebu	2629	948
Rissa	6543	622
Roan	994	375
Røros	5581	1956
Selbu	3996	1235
Skaun	6756	224
Snillfjord	992	508
Trondheim	173486	342
Tydal	886	1329
Ørland	5133	74
Åfjord	3229	955

Datastruktur

- En struktur for hver kommune
  - Felter
    - Navn
    - Innbyggere
    - Areal
- En vektor med strukturer for fylket
- Lager funksjon som leser fil-data inn i slik datastruktur
  - Inndata: Filnavn
  - Utdata: Vektor med strukturer



lesKommuneFil.m

```
function kommuner = lesKommuneFil(filNavn)

    fid = fopen(filNavn, 'r', 'native', 'utf8');
    % 'UTF-8' / 'latin1' / ASCII %% utf8 i Matlab-versjon R2013a !
    if fid == -1
        kommuner = -1;
    else
        indeks = 1;

        while ~feof(fid)
          % leser linje fra filen
          filLinje = fgetl(fid);
          % putter data inn i struktur
          kommuner(indeks) = lagPost(filLinje);
          % indeks for evt. neste post
          indeks = indeks + 1;
        end % while

        status = fclose(fid);
        if status ~= 0
           kommuner = -1;
        end
    end % if fid ok

end % function
```


lesKommuneFil.m

```
function post = lagPost(tekst)
% putter data for en kommune inn i struktur

    % plukker ut kommunenavn
    [post.navn, resten] = strtok(tekst);

    % plukker ut innbyggertall og areal
    tallData = str2num(resten);    % str2num('10 20') gir vektoren [10 20]
    post.innbyggere = tallData(1);
    post.areal = tallData(2);

end % function

function post = lagPost2(tekst)
% alternativ måte å gjore det

    % plukker ut kommunenavn
    [post.navn, resten] = strtok(tekst);

    % plukker ut innbyggertall
    [tall, resten] = strtok(resten);
    post.innbyggere = str2num(tall);

    % plukker ut areal
    [tall, resten] = strtok(resten);
    post.areal = str2num(tall);

end % function
```

```
>> kommuner = lesKommuneFil('datafil.txt')

kommuner =

1x25 struct array with fields:
    navn
    innbyggere
    areal

>> kommuner(1)

ans =

          navn: 'Agdenes'
    innbyggere: 1745
         areal: 318

>> kommuner(25)

ans =

          navn: 'Aafjord'
    innbyggere: 3229
         areal: 955

>>
```


Befolkningstetthet i Sør-Trøndelag

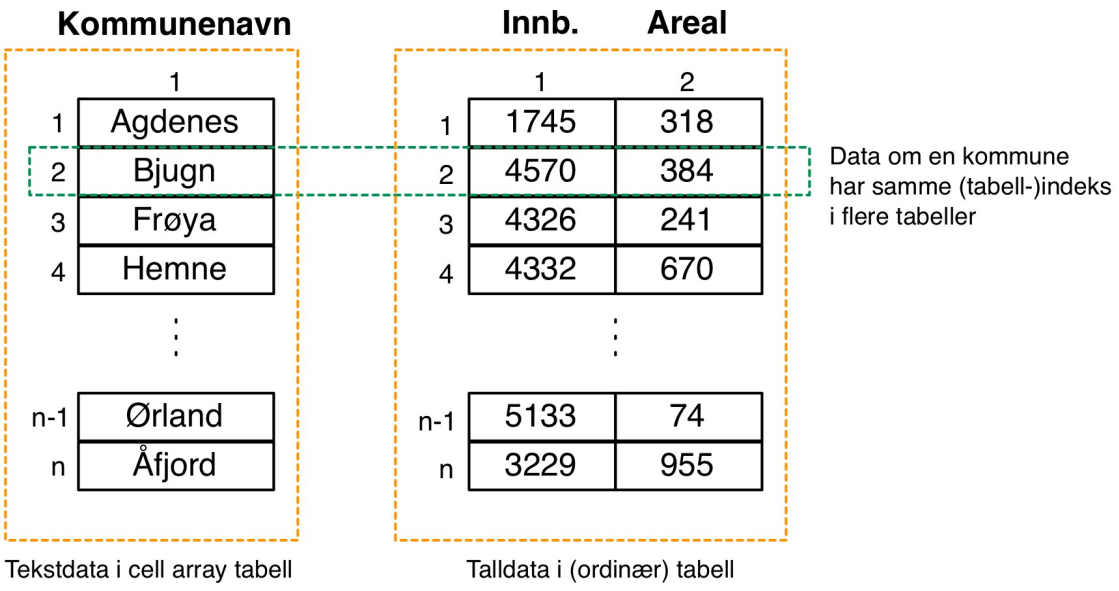
- Summere innbyggertall i alle kommuner
- Summere areal i alle kommuner
- Tetthet = sum(innbyggere) / sum(areal)
- Lager funksjon som gjør dette:
  - Inndata: Vektor med strukturer (kommunedataene)
  - Utdata: Befolkningstetthet

befolkningsTetthet.m

```
function tetthet = befolkningsTetthet(kommuner)

    % antall kommuner
    n = length(kommuner);

    % summerer befolkning og areal
    befolkning = 0;
    areal = 0;

    for i = 1:n
        befolkning = befolkning + kommuner(i).innbyggere;
        areal = areal + kommuner(i).areal;
    end % for

    % beregner befolkningstetthet
    tetthet = befolkning / areal;

end % function
```

```
>> t = befolkningsTetthet(k)

t =

   15.6027

>>
```


Alternativ løsning: Parallelle vektorer



```
>> [kNavn, kData] = lesKommuneFil_alt2('datafil.txt')
kNavn =
  Columns 1 through 7
    'Agdenes'    'Bjugn'    'Froya'    'Hemne'    'Hitra'    'Holtaalen'    'Klaebu'
  Columns 8 through 14
    'Malvik'    'Meldal'    'Melhus'    'Midtre-Gauldal'    'Oppdal'    'Orkdal'    'Osen'
  Columns 15 through 21
    'Rennebu'    'Rissa'    'Roan'    'Roros'    'Selbu'    'Skaun'    'Snillfjord'
  Columns 22 through 25
    'Trondheim'    'Tydal'    'Oerland'    'Aafjord'
kData =
        1745         318
        4570         384
        4326         241
        4232         670
        4340         686
        2048        1210
        5894         186
       12677         169
        3903         613
       15114         695
        6050        1861
        6691        2274
       11365         594
        1025         387
        2629         948
        6543         622
         994         375
        5581        1956
        3996        1235
        6756         224
         992         508
      173486         342
         886        1329
        5133          74
        3229         955
>> t = befolkningsTetthet_alt2(kData)
t =
   15.6027
>>
```


24
```
function [kNavn, kData] = lesKommuneFil_alt2(filNavn)

      fid = fopen(filNavn, 'r', 'native', 'utf-8');
      if fid == -1
          disp('Får ikke åpnet filen')
      else
          indeks = 1;

          while ~feof(fid)
            % leser linje fra filen
            filLinje = fgetl(fid);
            % plukker ut dataelementene
            [navn, resten] = strtok(filLinje);
            tallData = str2num(resten);
            % putter data inn i datastrukturene
            kNavn{indeks} = navn;
            kData(indeks, 1) = tallData(1);
            kData(indeks, 2) = tallData(2);
            % indeks for evt. neste post
              indeks = indeks + 1;
          end % while

          status = fclose(fid);
          if status ~= 0
             disp('Kan ikke lukke filen')
          end
      end % if

  end % function
```

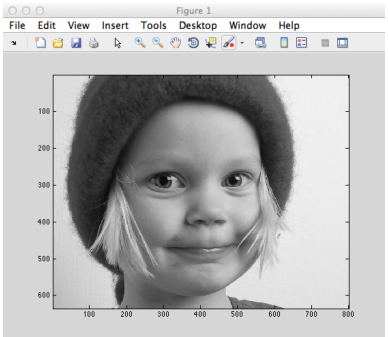
```
function tetthet = befolkningsTetthet_alt2(kommuneData)

    % beregner befolkningstetthet
    tetthet = sum(kommuneData(:,1)) / sum(kommuneData(:,2));

end % function
```

```
>> befolkningsTetthet_alt2(kData)

ans =

   15.6027

>>
```


Noen bildeeksempler

- Representasjon: To-dimensjonal tabell
  - Rader x kolonner med verdi for hver bildeelement (piksel)
- Gråtonebilder
  - 0 = sort, 255 = hvitt, gråtoner mellom
- Bilder har ”fargekart” som definerer hvilken farge hver pikselverdi skal ha.

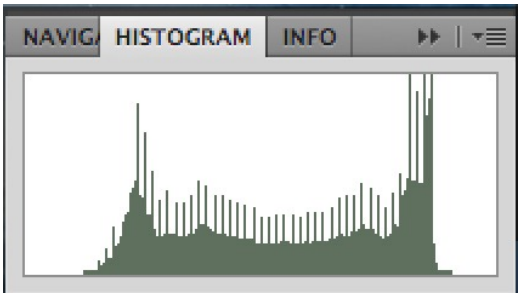
```
% leser inn bilde fra fil
[bilde, map] = imread('jente.gif');

% setter riktig "fargekart"
colormap(map)

% viser bildet
image(bilde)
```

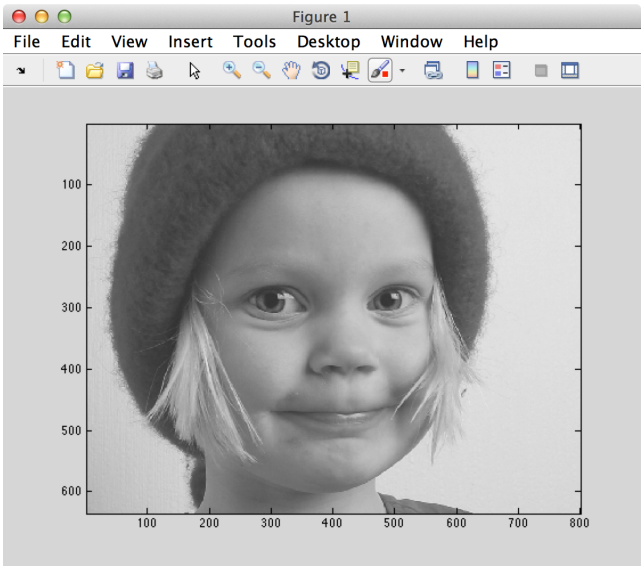


Problem: Lite kontrast

- Utnytter ikke toneomfanget
- Mørkeste piksel: 31
- Lyseste piksel: 233

```
>> [bilde, map] = imread('jente1.gif');
>> colormap(map)
>> image(bilde)
>>
```





NTNU

Løsning

- Skalerer 31-233 til 0-255
  - (pikselverdi-31)*255/(233-31)
    - 31 -> 0
    - 233 -> 255
- NB! Bildetabellene våre har datatype **uint8** (8 bit unsigned integer, dvs 0-255)
  - "Arithmetic operations that involve both integers and floating-point always result in an integer data type."
  - Det er lett å regne feil (se øverst til høyre)
- Typekonvertering (type cast)
  - B = cast(A,NEWCLASS)
  - minPiksel = cast(minPiksel,'double');
    - Gjør om ("31") til double-representasjon

```
>> a = uint8(130);
>>
>> b = a + 100
b =
  230
>> c = a + b
c =
  255
>> d = 2*a
d =
  255
>>
```

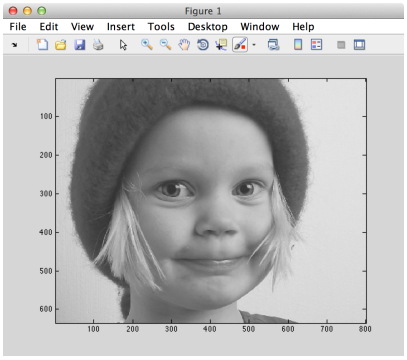
NTNU

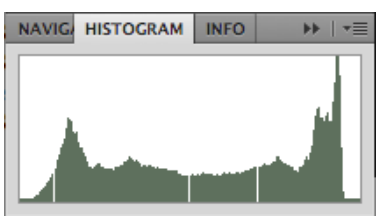
maksKontrast.m

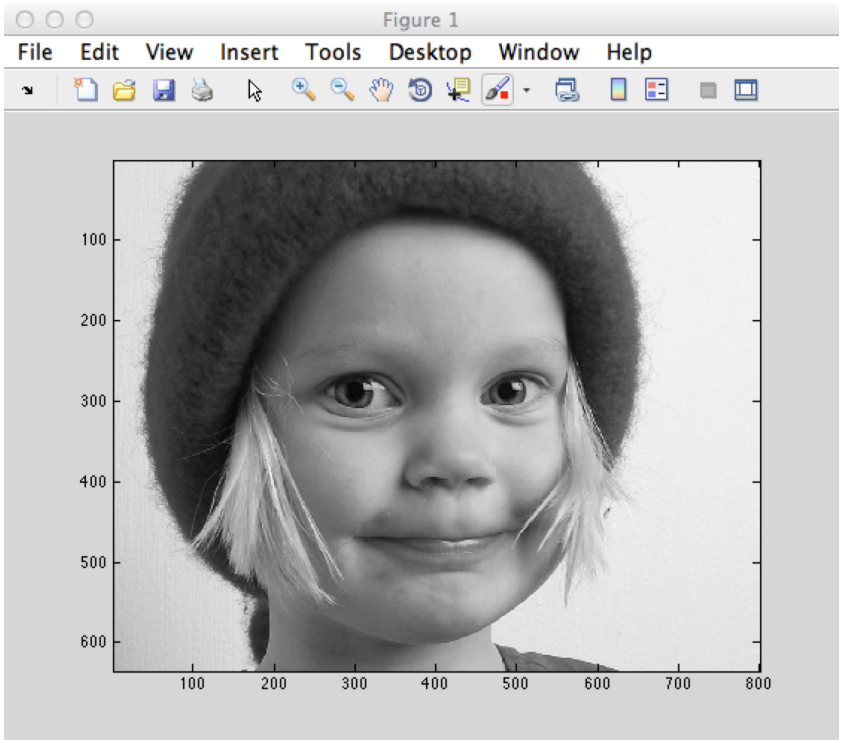
```
function bilde = maksKontrast(bilde)

    minPiksel = min(min(bilde));
    maksPiksel = max(max(bilde));

    minPiksel = cast(minPiksel,'double');
    maksPiksel = cast(maksPiksel,'double');

    strekkeFaktor = 255/(maksPiksel-minPiksel);

    bilde = round( (bilde-minPiksel)*strekkeFaktor );

end % function
```

```
>> nyttbilde = maksKontrast(bilde);
>> max(max(nyttbilde))
ans =
  255
>> min(min(nyttbilde))
ans =
    0
>> image(nyttbilde)
>>
```

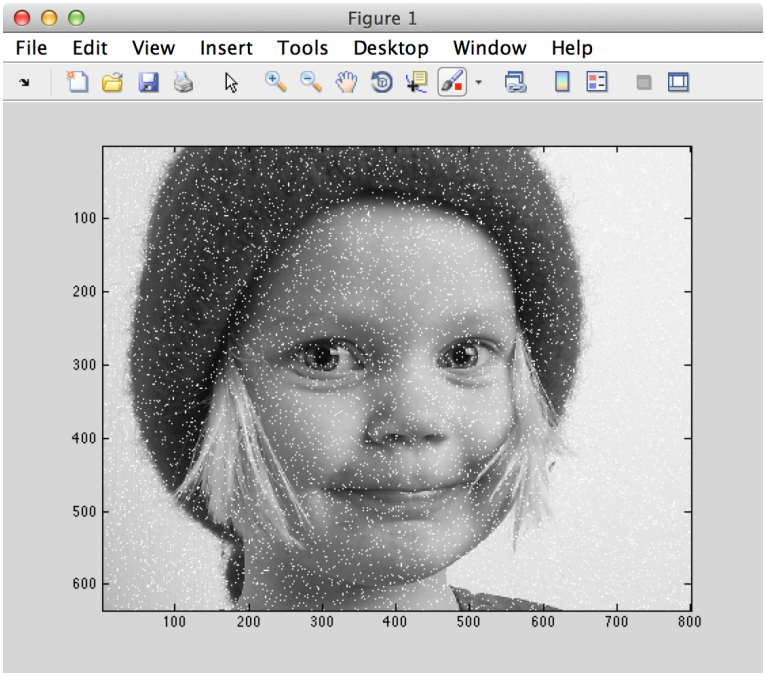

Resultat

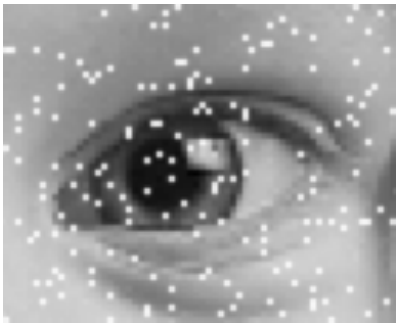






Bilde med støy (hvite piksler)

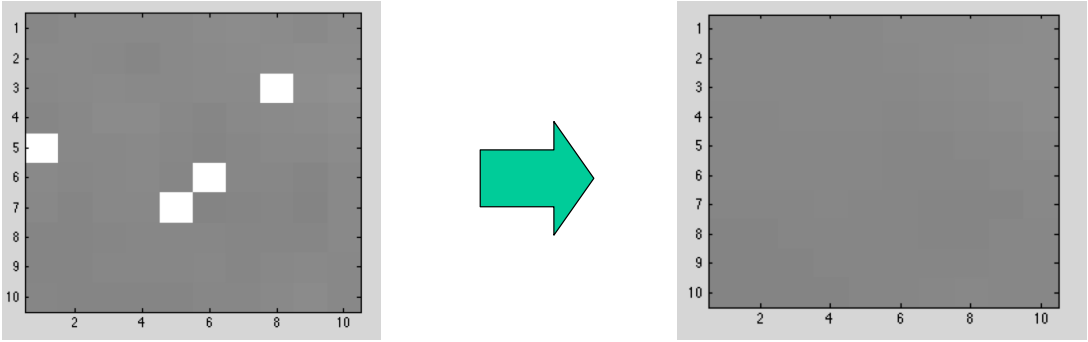




Filtrere støy med medianfilter

- Observasjon: Støypikslene skiller seg (stort sett) mye ut
- Idé: Erstatte støypikslene med en riktigere verdi fra omgivelsene
- Alle piksler har 8 naboer (unntatt kantpikslene som vi ignorerer)
- Får ”nabolag” på 3x3-piksler
  - Bruke gjennomsnittet av nabolaget (støypikslene ”trekker”)
  - Sortere pikselverdiene i nabolaget og bruke medianen (støypikslene blir marginalisert)
- Median-filter
  - Fjerner støy
  - Mister litt bildekvalitet (utenom støypikslene)

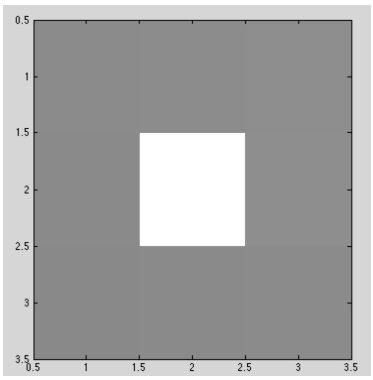
Utsnitt fra panna



116	118	118	118	118	120	121	120	118	121
118	118	117	115	118	119	120	121	122	122
117	118	119	118	118	119	119	255	123	125
115	117	120	120	117	115	118	119	120	122
255	117	117	118	116	114	116	118	118	120
118	116	117	118	115	255	116	116	114	118
117	114	117	117	255	113	114	115	113	118
114	114	115	116	116	116	115	115	115	118
114	114	116	116	116	117	115	117	118	117
115	114	114	114	114	116	116	118	120	117

118	118	118	118	118	120	120	120	121	122
118	118	118	118	118	119	120	121	122	122
118	118	118	118	118	118	119	120	122	122
117	117	118	118	118	117	118	119	120	122
117	117	117	117	117	116	116	118	118	120
117	117	117	117	117	116	116	116	118	118
116	116	116	117	116	116	115	115	115	118
114	114	116	116	116	116	115	115	117	117
114	114	114	116	116	116	116	116	117	117
114	114	114	114	116	116	117	118	117	117

Detalj (for en piksel)



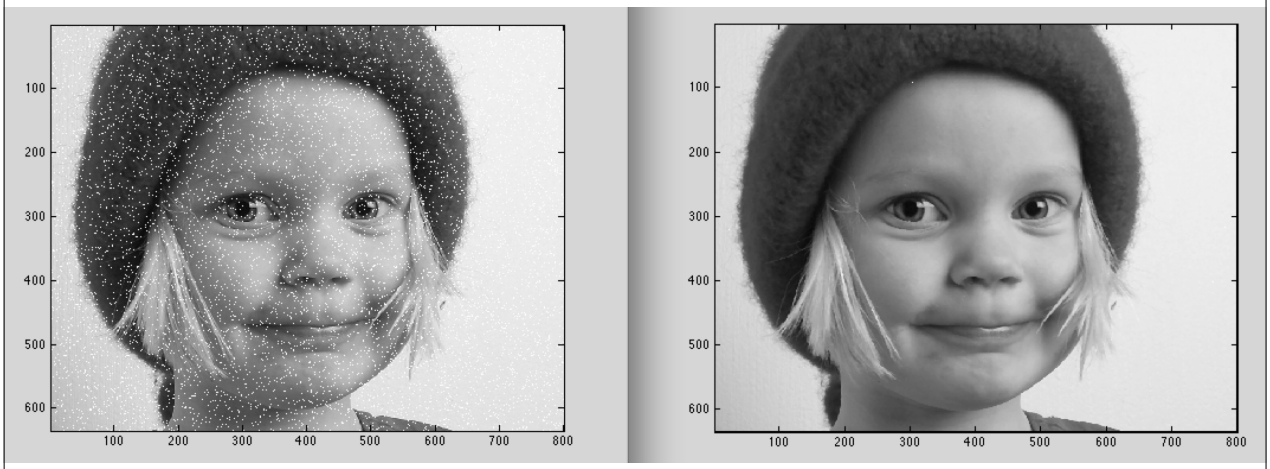
121	120	116
120	121	122
119	255	123
118	119	120
116	118	118

1	120		1	118
2	119		2	119
3	118		3	119
4	121		4	120
5	255	Sorterer	5	120
6	119		6	121
7	122		7	122
8	123		8	123
9	120		9	255

medianFilter.m

```
function bildeUt = medianFilter(bildeInn)

   [rader, kolonner] = size(bildeInn);

   bildeUt = zeros(rader, kolonner);

   for r = 2:rader-1
       for k = 2:kolonner-1

           bildeUt(r,k) = medianPiksel(bildeInn(r-1:r+1, k-1:k+1));

       end %for
   end %for

end % function

function piksel = medianPiksel(M)

    vektor = sort(M(:));
    piksel = vektor(5);

end % medianPiksel
```


Resultat



Kant-filter

- Ser på nabolaget rundt hver piksel (3x3)
- En maske for å oppdage "horisontale" kanter
  - $Gx = (z_7+2z_8 + z_9) - (z_1 + 2z_2 + z_3)$

z1	z2	z3
z4	z5	z6
z7	z8	z9

- En maske for å oppdage "vertikale" kanter
  - $G_y = (z_3+2z_6 + z_9) - (z_1 + 2z_4 + z_7)$

-1	-2	-1
0	0	0
1	2	1

- Kantstyrken i hvert punkt
  - $G = |G_x| + |G_y|$
- Dette kalles en Sobel-operator
  - Finne alternative måter å finne kanter

-1	0	1
-2	0	2
-1	0	1

Eksempel

```
>> bildebit
bildebit =
     0     0     0     0     0     0
     0     9     9     9     9     0
     0     9     9     9     9     0
     0     9     9     9     9     0
     0     0     0     0     0     0
>> kanter = kantFilter(bildebit)
kanter =
     0     0     0     0     0     0
     0    54    36    36    54     0
     0    36     0     0    36     0
     0    54    36    36    54     0
     0     0     0     0     0     0
>>
```

39

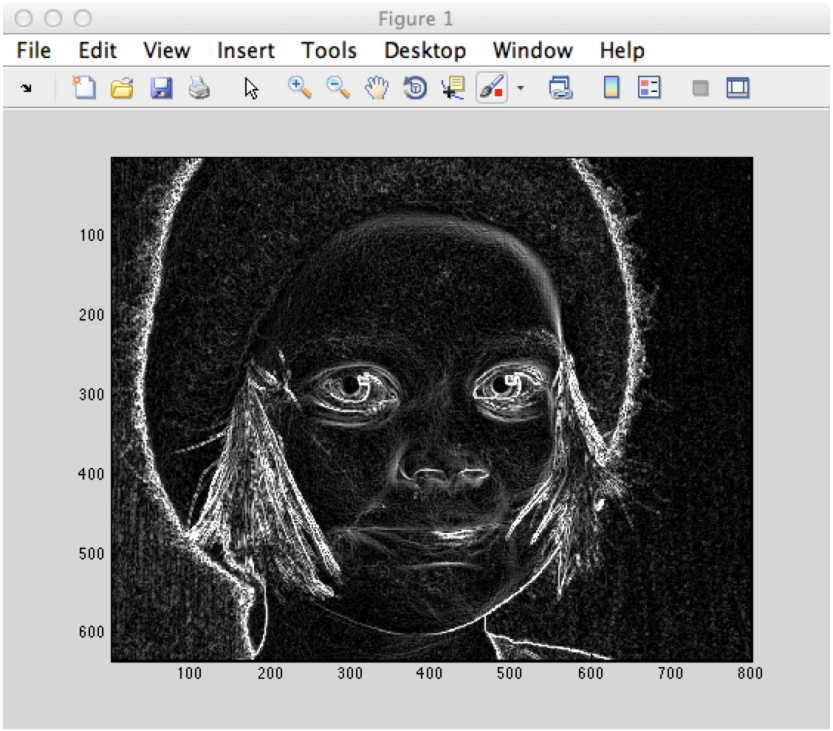
kantFilter.m

```
function bildeUt = kantFilter(bildeInn)

    [rader, kolonner] = size(bildeInn);

    bildeInn = cast(bildeInn, 'double');
    bildeUt = zeros(rader, kolonner);

    for r = 2:rader-1
        for k = 2:kolonner-1

            bildeUt(r,k) = sobelOperator(bildeInn(r-1:r+1, k-1:k+1));

        end %for
    end %for

end % function

function gradient = sobelOperator(M)

     gX = -M(1,1) - 2*M(1,2) - M(1,3) + M(3,1) + 2*M(3,2) + M(3,3);
     gY = -M(1,1) - 2*M(2,1) - M(3,1) + M(1,3) + 2*M(2,3) + M(3,3);

     gradient = abs(gX) + abs(gY);

end % function
```

NTNU

40

Resultat



NTNU

NTNU

Kunnskap for en bedre verden

Kunnskap for en bedre verden

Eksamenseksempel
Konte - aug 2015

Amanuensis Terje Rydland
Kontor: ITV-021 i IT-bygget vest (Gløshaugen)
Epost: terjery@idi.ntnu.no
Tlf: 735 91845

Konte - aug 2015

Oppgave 3 Programmering Yatzy (15%)

Du kan anta at alle funksjonene mottar gyldige argumenter (inn-verdier).

I denne oppgaven skal du skrive noen funksjoner til spillet Yatzy. I Yatzy spiller man med 5 terninger der målet er å få høyest poengsum på ulike sammensetninger av terninger som mange like, ett par, to par, tre like, fire like, liten straight, stor straight, hus, sjanse og yatzy.

Oppgave 3 a) (3%)

Lag funksjonen **throw** som har inn-parameteren **n**. Funksjonen skal returnere en liste med **n** tilfeldige verdier mellom 1 og 6.

Oppgave 3 b) (3%)

Lag funksjonen **chance** som har inn-parameteren **dice** som er en liste med n elementer med verdier mellom 1 og 6. Funksjonen skal returnere summen av alle elementene i lista og skal lages uten ved hjelp av innebygde funksjoner for å summere lister.

Oppgave 3 c) (4%)

Lag funksjonen **house** som har inn-parameteren **dice** som er en liste med fem elementer med verdier mellom 1 og 6. Funksjonen skal returnere summen av alle terningene hvis verdiene i **dice** har både 3 like og ett par (f.eks. 4,4,4,2,2 eller 1,1,6,6,6), hvis ikke skal verdien 0 returneres. Funksjonen skal også returnere verdien 0 hvis alle elementene i **dice** er like.

Oppgave 3 d) (5%)

Lag funksjonen **straight** som har inn-parameteren **dice** som er en liste med fem elementer med verdier mellom 1 og 6. Funksjonen skal undersøke om listen representerer en liten eller stor straight. Følgende skal returneres:

For en liten straight (liste som inneholder tallene 1,2,3,4,5) skal funksjonen returnere tallet 15. For en stor straight (liste som inneholder tallene 2,3,4,5,6) skal funksjonen returnere tallet 20. For en liste som er verken liten eller stor straight skal funksjonen returnere tallet 0.

43

Oppgave 3 a) (3%)

Lag funksjonen **throw** som har inn-parameteren **n**. Funksjonen skal returnere en liste med **n** tilfeldige verdier mellom 1 og 6.

```
>> throw(5)
 ans =
     4     1     6     6     5
```

```
function out = throw(n)
    out = randi(6,1,n);
end %function
```


44

Oppgave 3 b) (3%)

Lag funksjonen **chance** som har inn-parameteren **dice** som er en liste med n elementer med verdier mellom 1 og 6. Funksjonen skal returnere summen av alle elementene i lista og skal lages uten ved hjelp av innebygde funksjoner for å summere lister.

```
>> dice = [5, 2, 6, 3, 3];
>> chance(dice)
ans =    19
```

```
function out = chance( dice )
    out = 0;
    for i = 1:length( dice )
        out = out + dice(i);
    end %for
end %function
```

Oppgave 3 c) (4%)

Lag funksjonen **house** som har inn-parameteren **dice** som er en liste med fem elementer med verdier mellom 1 og 6. Funksjonen skal returnere summen av alle terningene hvis verdiene i **dice** har både 3 like og ett par (f.eks. 4,4,4,2,2 eller 1,1,6,6,6), hvis ikke skal verdien 0 returneres. Funksjonen skal også returnere verdien 0 hvis alle elementene i **dice** er like.

```
>> house([1,3,1,1,3])
ans =
     9
>> house([4,3,3,3,4])
ans =
    17
>> house([2,2,2,2,2])
ans =
     0
>> house([1,3,4,2,3])
ans =
     0
```

```
function out = house( dice )
    out = 0;
    dice = sort(dice);
    if ( dice(1) ~= dice(5) )
        if ( dice(1) == dice(3) && dice(4) == dice(5) ...
                || dice(1) == dice(2) && dice(3) == dice(5) )
            out = sum(dice);
        end %if 3+2 OR 2+3 equal
    end %if not all equal
end %function
```


Oppgave 3 d) (5%)

Lag funksjonen **straight** som har inn-parameteren **dice** som er en liste med fem elementer med verdier mellom 1 og 6. Funksjonen skal undersøke om listen representerer en liten eller stor straight. Følgende skal returneres:

For en liten straight (liste som inneholder tallene 1,2,3,4,5) skal funksjonen returnere tallet 15. For en stor straight (liste som inneholder tallene 2,3,4,5,6) skal funksjonen returnere tallet 20. For en liste som er verken liten eller stor straight skal funksjonen returnere tallet 0.

```
>> straight( [4,2,1,3,5] )
ans =
    15
>> straight( [5,6,4,2,3] )
ans =
    20
>> straight( [1,1,2,3,3] )
ans =
     0
```

```
function out = straight2( dice )
    dice = sort(dice);
    out = 0;
    liten = [1 2 3 4 5];
    stor = [2 3 4 5 6];
    if isequal(dice,liten)
        out = 15;
    elseif isequal(dice,stor)
        out = 20;
end %function
```

```
function out = straight( dice )
    dice = sort(dice);
    out = dice(1);
    if ( out == 1 || out == 2 )
        for i = 2:5
            if ( dice(i-1) == dice(i)-1 )
                out = out + dice(i);
            else
                out = 0;
                return %abort if not directly following previous die value
            end % if actual straight
        end %for all neighbors
    end % if possible straight
end %function
```

Konte - aug 2015

Oppgave 4 Programmering vitneobservasjoner (45%)

I noen av oppgavene 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.

Politiet trenger et system for å sjekke om vitneobservasjoner av kjøretøyer fra hendelser som etterforskes, stemmer med faktiske kjøretøyer i et register.

Oppgave 4 a) (5%)

Skriv en funksjon **les_inn_bilinfo( )** som leser inn fra tastatur vitnets observasjon av bilmerke, modell og farge for et kjøretøy. Funksjonen skal returnere disse tre opplysningene i en liste.

Oppgave 4 b) (5%)

Skriv en funksjon **sjekk_bil(witness, car)** som sammenligner to lister som hver inneholder tre tekststrenger, der den ene lista representerer en vitneobservasjon og den andre en faktisk bil. I vitneobservasjonen kan felt inneholde '?' som betyr at vitnet var usikker på den informasjonen. Funksjonen skal returnere `true` (1) eller `false` (0), `true` hvis det er full match eller hvis avvik kun gjelder '?', `false` hvis det fins avvik som ikke er '?'.

Oppgave 4 c) (5%)

Gitt strengen `SKILTBOKSTAV = 'ABCDEFGHJKLNPRSTUVXYZ?';`
Den inneholder bokstaver som er lov å bruke på moderne norske bilskilt, samt '?' helt bakerst (hvis vitnet ikke husker). Skriv en funksjon **les_gyldig_vitneskilt()** som leser inn fra tastatur en streng på nøyaktig 7 tegn, hvorav de 2 første tegnene skal være tegn fra strengen SKILTBOKSTAV, og de fem siste tegnene skal være tall eller '?'. Ved feil input skal funksjonen be brukeren gjøre et nytt forsøk, inntil input er gyldig. Da skal funksjonen returnere strengen.

Oppgave 4 d) (5%)

Skriv en funksjon **match(witness, car)** som skal sjekke om et vitneobservert skilt kan stemme overens med et faktisk skiltnummer. Funksjonen må ta inn de to strengene som skal sammenlignes som parametere. Returner `true` (1) hvis det er en hel match (strengene er identiske) eller mulig match (de eneste forskjellene skyldes '?'), og `false` (0) hvis de to ikke kan stemme overens.

Konte - aug 2015

Oppgave 4 Programmering vitneobservasjoner (45%)

Oppgave 4 e) (5%)

Skriv en funksjon **match_liste(witness, liste)** som sammenligner ett vitneobservert skilt med en liste av faktiske skilt. Funksjonen skal returnere lista av alle skilt som *kan* stemme med det observerte skiltet.

Oppgave 4 f) (20%)

Anta at vi har en tekstfil `biler.txt` med format som vist i utdraget under, dvs. skiltnummer, bilmerke, modell, farge og navn på eier, hvor hvert element er adskilt med mellomrom.

Skriv et skript eller en main()-funksjon som gjør følgende:

- Leser inn data fra fila `biler.txt` og putter i en data-struktur. Bruk unntaksbehandling for å unngå krasj hvis fila mangler.
- La brukeren sjekke den ene vitneobserverte bilen etter den andre opp mot det som fins i datastrukturen, inntil brukeren ønsker å slutte.
- For hver bil som sjekkes, skriv ut potensielle treff til skjerm, dvs. alle biler hvor de opplysningene som ikke var '?', matchet. Vis på skjerm skiltnummer og navn på eier.
- Hvis ingen kjøretøy matcher, skal programmet skrive ut 'Ingen match'

```
DK21518 FIAT Panda Blå Os,Liss
GH70709 Ford Mondeo Blå Hansen,Jo
FB37769 FIAT Panda Brun Å,Ole
TD79641 Ford S-Max Grå Berg,Jo
PE66975 Toyota Avensis Gul Nes,Al
JV13133 VW Polo Brun Bø,Ole
CG74083 FIAT Panda Blå Hansen,Ann
ZG27056 Toyota Previa Grønn Berg,Bo
```

Du bestemmer selv om du vil skrive all koden for dette i skriptet / main( ), eller om du vil dele det opp i flere funksjoner, men god oppdeling vil telle positivt der det er naturlig. Likeledes vil det telle positivt om du klarer å bruke funksjoner fra tidligere deloppgaver der det passer.

Konte - aug 2015

Oppgave 4 a) (5%)

Skriv en funksjon **les_inn_bilinfo( )** som leser inn fra tastatur vitnets observasjon av bilmerke, modell og farge for et kjøretøy. Funksjonen skal returnere disse tre opplysningene i en liste.

```
>> les_inn_bilinfo()
Hvilket bilmerke var det? FIAT
Hvilken modell? Uno
Hvilken farge? Rød
ans =
    'FIAT'    'Uno'    'Rød'
```

```
function liste = les_inn_bilinfo( )
   liste{1} = input( 'Hvilket bilmerke var det? ', 's' );
   liste{2} = input( 'Hvilken modell? ', 's' );
   liste{3} = input( 'Hvilken farge? ', 's' );
end %function
```


Konte - aug 2015

Oppgave 4 b) (5%)

Skriv en funksjon **sjekk_bil(witness, car)** som sammenligner to lister som hver inneholder tre tekststrenger, der den ene lista representerer en vitneobservasjon og den andre en faktisk bil. I vitneobservasjonen kan felt inneholde '?' som betyr at vitnet var usikker på den informasjonen. Funksjonen skal returnere `true` (1) eller `false` (0), `true` hvis det er full match eller hvis avvik kun gjelder '?', `false` hvis det fins avvik som ikke er '?'.

```
>> sjekk_bil({'FIAT','Uno','Rød'},{'FIAT','Uno','Rød'})
ans =
     1
>> sjekk_bil({'FIAT','Uno','Rød'},{'FIAT','Uno','Blå'})
ans =
     0
>> sjekk_bil({'FIAT','Uno','?'},{'FIAT','Uno','Rød'})
ans =
     1
>> sjekk_bil({'FIAT','Uno','?'},{'FIAT','Punto','Rød'})
ans =
     0
```

```
function match = sjekk_bil( vitne, bil )
   match = true;
   for i = 1:3
   if (~( strcmp(vitne{i},'?') || strcmp(vitne{i}, bil{i})))
        match = false;
    end
end %function
```

Konte - aug 2015

Oppgave 4 c) (5%)

Gitt strengen `SKILTBOKSTAV = 'ABCDEFGHJKLNPRSTUVXYZ?';`
Den inneholder bokstaver som er lov å bruke på moderne norske bilskilt, samt '?' helt bakerst (hvis vitnet ikke husker). Skriv en funksjon **les_gyldig_vitneskilt()** som leser inn fra tastatur en streng på nøyaktig 7 tegn, hvorav de 2 første tegnene skal være tegn fra strengen SKILTBOKSTAV, og de fem siste tegnene skal være tall eller '?'. Ved feil input skal funksjonen be brukeren gjøre et nytt forsøk, inntil input er gyldig. Da skal funksjonen returnere strengen.

```
function wnr = les_gyldig_vitneskilt()
   read_more = true;
   while (read_more)
      wnr = input('Skriv inn skilt, 2 bokst + 5 tall (?=usikker) ', 's');
      if ( length( wnr ) ~= 7 )
         disp('Skiltnummer må være 7 tegn langt');
      elseif (~(match_bokstav(wnr(1)) && match_bokstav(wnr(2))))
         disp('To første tegn må være lovlig skiltbokstav eller ?')
      else
         read_more = false; %Anta tallene er riktig
         for i=3:7
            if (~strcmp(wnr(i), '?' ) && isempty(sscanf(wnr(i),'%d')))
               disp( 'Fem siste tegn må være tall eller ?' )
               read_more = true;
               break; %for 5 tall, ikke skriv mer enn en feilmelding
            end % if tegn i NOT ? or Tall
         end % for tall 3 til 7
      end % if noen feil, else OK
   end % while noen feil
end %function
```


Konte - aug 2015

Oppgave 4 c) (5%)

Gitt strengen `SKILTBOKSTAV = 'ABCDEFGHJKLNPRSTUVXYZ?';`
Den inneholder bokstaver som er lov å bruke på moderne norske bilskilt, samt '?' helt bakerst (hvis vitnet ikke husker). Skriv en funksjon **les_gyldig_vitneskilt()** som leser inn fra tastatur en streng på nøyaktig 7 tegn, hvorav de 2 første tegnene skal være tegn fra strengen SKILTBOKSTAV, og de fem siste tegnene skal være tall eller '?'. Ved feil input skal funksjonen be brukeren gjøre et nytt forsøk, inntil input er gyldig. Da skal funksjonen returnere strengen.

```
function match = match_bokstav(c)
   SKILTBOKSTAV = 'ABCDEFGHJKLNPRSTUVXYZ?'; %Oppgitt i oppgaveteksten
   match = false;
   for i = 1:length(SKILTBOKSTAV)
      if strcmp(c, SKILTBOKSTAV(i) )
         match = true;
       end %if match
    end %for all valid
end %helper function
```

Konte - aug 2015

Oppgave 4 d) (5%)

Skriv en funksjon **match(witness, car)** som skal sjekke om et vitneobservert skilt kan stemme overens med et faktisk skiltnummer. Funksjonen må ta inn de to strengene som skal sammenlignes som parametere. Returner `true` (1) hvis det er en hel match (strengene er identiske) eller mulig match (de eneste forskjellene skyldes '?'), og `false` (0) hvis de to ikke kan stemme overens.

```
>> match('VF12345','VF12355')
ans =
     0
>> match('V?1234?','VF12355')
ans =
     0
>> match('VF???55','VF12355')
ans =
     1
>> match('??12355','VF12355')
ans =
     1
>> match('???????','VF12355')
ans =
     1
```

```
function hit = match( witness, car ) %antar korrekt input
   hit = true; %antar treff
   for i=1:7
      if (~(strcmp(witness(i),'?') || strcmp(witness(i),car(i))))
         hit = false;
      end % if
   end % for each symbol
end %function
```


Konte - aug 2015

Oppgave 4 e) (5%)

Skriv en funksjon **match_liste(witness, liste)** som sammenligner ett vitneobservert skilt med en liste av faktiske skilt. Funksjonen skal returnere lista av alle skilt som *kan* stemme med det observerte skiltet.

```
>> match_liste('VF???55',{'VX33322','VF12355','VF77455','DA?????','VF10055'})
ans =
    'VF12355'    'VF77455'    'VF10055'
```

```
function hits = match_liste( witness, list )
  hits = {};
  for i = 1:length(list)
    if match(witness,list{i} )
      hits{length(hits)+1} = list{i}
    end %if match
  end % for each row
end %function
```

Konte - aug 2015

Oppgave 4 f) (20%)

Anta at vi har en tekstfil `biler.txt` med format som vist i utdraget under, dvs. skiltnummer, bilmerke, modell, farge og navn på eier, hvor hvert element er adskilt med mellomrom.

Skriv et skript eller en main()-funksjon som gjør følgende:

- Leser inn data fra fila `biler.txt` og putter i en data-struktur. Bruk unntaksbehandling for å unngå krasj hvis fila mangler.
- La brukeren sjekke den ene vitneobserverte bilen etter den andre opp mot det som fins i datastrukturen, inntil brukeren ønsker å slutte.
- For hver bil som sjekkes, skriv ut potensielle treff til skjerm, dvs. alle biler hvor de opplysningene som ikke var '?', matchet. Vis på skjerm skiltnummer og navn på eier.
- Hvis ingen kjøretøy matcher, skal programmet skrive ut 'Ingen match'

```
DK21518 FIAT Panda Blå Os,Liss
GH70709 Ford Mondeo Blå Hansen,Jo
FB37769 FIAT Panda Brun Å,Ole
TD79641 Ford S-Max Grå Berg,Jo
PE66975 Toyota Avensis Gul Nes,Al
JV13133 VW Polo Brun Bø,Ole
CG74083 FIAT Panda Blå Hansen,Ann
ZG27056 Toyota Previa Grønn Berg,Bo
```

Du bestemmer selv om du vil skrive all koden for dette i skriptet / main( ), eller om du vil dele det opp i flere funksjoner, men god oppdeling vil telle positivt der det er naturlig. Likeledes vil det telle positivt om du klarer å bruke funksjoner fra tidligere deloppgaver der det passer.


```
fid = fopen('biler.txt') ;
if fid==-1
   disp('File open not sucessfull: biler.txt');
else
   cardata = textscan(fid,'%s %s %s %s %s');
   fprintf('File read\n');
      answer = 'Y';
      while (answer == 'Y')
      carinfo = les_inn_bilinfo() ;
      skilt = les_gyldig_vitneskilt() ;
      found = 0 ;
      for i=1:size(cardata{1},1)
         if sjekk_bil(carinfo,{cardata{2}{i},cardata{3}{i},cardata{4}{i}}) && ...
               match(skilt,cardata{1}{i})
            fprintf('%s Owner: %s\n', cardata{1}{i}, cardata{5}{i});
            found = 1 ;
               end %if
         end %for
      if ~found
         fprintf('No match\n') ;
      end %if
      answer = input('Do you want to continue (Y/N)? ', 's');
   end %while
end %if
```