Norges teknisk–naturvitenskapelige universitet
Institutt for datateknikk og informasjonsvitenskap

TDT4105 IT Grunnkurs
Høst 2014

Øving 2

I denne øvingen skal vi se nærmere på if-setninger, funksjoner, logiske utrykk, matriser og vektorer. Disse er alle områder som vil bli bygget videre på og som er grunnleggende i programmering. Alle teorispørsmål skal besvares og begrunnes, all kode skal skrives i interaktiv modus i Matlab med mindre annet er spesifisert i oppgaven. Alle oppgavene skal demonstreres til studass på sal. I oppgaver hvor du skal lage funksjoner eller skript i separate filer skal også filene vises frem. Lykke til!

1 Teori

a) Nevn to forskjeller på RAM og Harddisk.

b) Hva er den viktigste forskjellen på RAM og ROM?

c) Hva er fordelen med tilfeldig aksess sammenliknet med sekvensiell aksess?

d) Gitt følgende funksjon

```
function y =  identity(x)
  y = x;
end
```

1. Hva er `y`, `identity` og `x`?
2. Hva skjer på linje nummer 2?

2 Kodeforståelse

I denne oppgaven skal du lese kode og finne ut hva skriptene vil skrive ut på skjermen. Her er det viktig at du ikke tester ut koden før du har gjort deg opp en mening om hva utskriften er. Denne typen kodeforståelse er viktig for å lære seg å programmere og på eksamen. Hvis du ikke forstår hvordan koden fungerer etter å ha kjørt den, kan du spørre en studass om hjelp til å bruke debuggeren i MATLAB for å se hva koden gjør.

a)

```
a = 10;
b = 4;
c = 6;

if(b < a)
  a = 9 - b;
  c = 4;
else
  b = b - a;
  c = 15;
end

disp (a);
disp (b);
disp (c);
```

b)

```
a = 8;
b = 4;
c = 9;

if(b + c > a && c <= 12)
  b = a;
  a = b;
  if(a > b)
    c = 0;
  elseif (a < b)
    c = c + 1;
  else
    c = c + 2;
  end
else
  c = b;
end

disp (a);
disp (b);
disp (c);
```

c) Denne deloppgaven inneholder både en funksjon og et skript.

Funksjon:

```
function resultat = foo(a, x)
  resultat = a^x
end
```

Skript

```
a = 2;
b = 3;
c = foo(a, b);
disp(c);
```

Når du skal teste denne koden må du lage en funksjon. For å lage en funksjon må du lage en ny .m-fil. Det er viktig å gi filen det samme navnet som funksjonsnavnet. For å gjøre dette i MATLAB velger du File > New > Function. Da får du opp et editor-vindu hvor du kan endre og lagre funksjonen.

For å lage funksjonen i denne oppgaven må du kopiere koden og lime den inn over det som er i editoren, for så å lagre den som foo.m. Når du har gjort dette vil funksjonen være tilgjengelig i interaktiv modus, inne i andre funksjoner og i skript.

Du kan også lagre skriptet som en egen fil. Dette gjør du ved å velge File > New > Script. Skriptet kan nå kjøres ved å skrive filnavnet du valgte når du lagret i interaktiv modus og tykke enter.

3 Funksjoner og if-setninger

I denne oppgaven skal du lage fire funksjoner som til sammen regner ut boten man får for å kjøre for fort. Inputen til oppgaven er en fartsgrense samt bilens fart, og output er prisen på boten. Utregningen av boten tar først utgangspunkt i fartsgrensen, og deretter hvor mange km/t over fartgrensen bilen ble registrert. Tabell 1 viser de gjeldende satsene.

Tabell 1: Fartsgrenser og bøtesatser

Fartsgrense [km/t]	Overskridelse [km/t]	[km/t]
0 - 60	0 - 15	2900
	16 - ∞	6500
70 - 80	0 - 15	2600
	16- 25	4900
	26 - ∞	7800
90 - ∞	0 - 15	2600
	16 - 25	4900
	26 - 35	7800
	36 - ∞	9000

Hvis man for eksempel kjører 66 km/t i en 50-sone får man 6500 kr i bot. For å løse et sammensatt problem er det viktig å dele det opp i flere mindre problemer. I denne oppgaven kan man se to mindre problemer, hvor ett igjen har tre deler:

1. Avgjør hvilken fartsgrensekategori som gjelder
2. Gitt en fartgrensekategori og en fartsoverskridelse, avgjør hvilken bot som gjelder.
   - for 0 - 60
   - for 70 - 80
   - for 90 - ∞

Når man har identifisert problemene så bør man løse ett om gangen. Når man har løst alle delene kan man sette del-løsningene sammen til en løsning på hele problemet.

a) Først skal du lage en funksjon calcLow. Denne skal lagres i filen calcLow.m. I denne oppgaven antar vi at fartsgrensen er i kategorien 0 - 60, så vi trenger derfor kun å konsentrere oss om de to første radene i tabellen. Funksjonen skal ta inn som parameter hvor mye over fartsgrensen bilen kjørte og returnere prisen på boten. Det betyr at calcLow.m filen kan se slik ut i begynnelsen:

```
function ticket = calcLow( speedAboveLimit )
  % skriv kode her
end
```

Når du er ferdig kan du teste funksjonen fra interaktiv modus slik:

```
calcLow (0) % skal skrive ut 0
calcLow (1) % skal skrive ut 2900
calcLow (15) % skal skrive ut 2900
calcLow (16) % skal skrive ut 6500
```

b) Nå skal du lage funksjonen calcMedium. Denne skal også lagres i en egen fil. Nå antar vi at fartsgrensen er 70 - 80. Denne funksjonen tar inn farten over fartsgrensen som parameter og returnerer prisen på boten.

Denne funksjonen kan testes med følgende setninger.

```
calcMedium (0) % skal skrive ut 0
calcMedium (1) % skal skrive ut 2600
calcMedium (15) % skal skrive ut 2600
calcMedium (16) % skal skrive ut 4900
calcMedium (25) % skal skrive ut 4900
calcMedium (26) % skal skrive ut 7800
```

c) Lag så funksjonen calcHigh. Her antar vi at fartsgrensen er 90 eller høyere. Denne funksjonen skal ta inn farten over fartsgrensen som paramater og returnere prisen på boten.

Hvilke setninger må du skrive for å teste alle utfallene for denne funksjonen?

Nå har du løst alle de små delproblemene. Du skal i denne oppgaven sette sammen løsningene for å løse hele problemet. Lag en funksjon, calcSpeedingTicket, som tar inn farten til bilen og fartsgrensen som parameter. Denne funksjonen skal også returnere prisen. For å returnere riktig pris må funksjonen først avgjøre hvilken kategori fartsgrensen tilhører. Deretter må den kalle på riktig funksjon (fra oppgave a, b og c) med riktig parameter og returnere verdien den får fra denne funksjonen.

Tips: Siden denne funksjonen ikke får inn differansen mellom fartsgrensen og farten, må du regne den ut før du kaller på rett funksjon.

4 Matlab-drill

I denne oppgaven skal vi jobbe med matriser og vektorer. For å kunne hente ut og/eller endre på elementene i en matrise må vi ha en måte å referere til hver element i matrisen på. Dette gjør vi ved å angi en rad og en kolonne på følgende måte. Gitt en matrise med navn m vil m(1,1) gi tilgang til elementet på rad og kolonne en. Lag et nytt skript og definer følgende to matriser:

```
m1 = [ 1:5; 6:10; 11:15; 16:20; 21:25 ];
m2 = ones (5, 5);
```

Funksjonen ones lager en matrise med 1 som verdi for alle elementene. Linjene over lager to 5x5-matriser.

a) Sett elementet nederst til venstre i m1 inn på plassen øverst til venstre i m2.

Hvis man vil ha tilgang til en hel rad eller kolonne i en matrise kan man bruke : i indeks operatoren. m(:,1) vil gi tilgang til hele den første kolonnen i matrisen m.

b) Lag en vektor v1 og sett den lik verdien til første rad i m1.

c) Kopier verdiene fra v1 til rad 2 i m2.

d) Kopier verdiene fra rad 3 i m1 til kolonne 3 i m2.

Hvis du har utført alle operasjonene riktig skal matrisen m2 se slik ut:

```
m2 =

    21     1    11     1     1
     1     2    12     4     5
     1     1    13     1     1
     1     1    14     1     1
     1     1    15     1     1
```