# Løsningsforslag til ordinær eksamen i TDT4105 Informasjonsteknologi grunnkurs (MATLAB)

Oppgave 1-20 (multiple choice)
Korrekte svaralternativer er:

1. 4011
2. Sampling
3. 6D726B6D ***(men se under)***
4. Nødvendig samplingsrate i forhold til høyeste frekvens som […]
5. At alle elementer kan hentes direkte.
6. Transistorer
7. Solid state drive
8. Det er en halvleder
9. Utføre regneoperasjoner
10. En alternativ form for maskinspråk som bruker […]
11. En teknikk der en CPU kan utføre flere instruksjoner parallelt.
12. 1 byte
13. Wide Area Network
14. 16
15. TCP
16. ACK (retransmisjon)
17. Internet of Things
18. De er oftest innebygd i switcher og rutere.
19. At en privat nøkkel brukes til å kryptere en melding.
20. Kryptering

Oppgave 3 hadde ingen korrekte svaralternativ. Fasitsvaret var «6D726B6D» men disse fire tegnene er ikke «OSLO» tolket som ASCII. Denne oppgaven fjernes derfor fra eksamen. For å gjøre det enkelt å gjennomføre, gis det to poeng til alle studenter på denne oppgaven – det var bare 15 studenter som ikke hadde svart eller hadde svart noe annet enn fasitsvaret på denne oppgaven.

I multiple choice-delen talte hver oppgave 2 poeng ved rett svar og 0 poeng ved feil eller dersom det ikke var avgitt noe svar. I de øvrige oppgavene talte besvarelsen fra 0 til 10 poeng.

## Oppgave 21-24 (Kodeforståelse)

### Oppgave 21 (Største felles multiplum)

Denne oppgaven regner ut største felles faktor mellom to tall, og mellom 42 og 14 er største felles faktor **14**. Mellom 42 og 35 ville største felles faktor være 7. Som rett svar på beskrivelse er

godkjent **«regner ut største felles faktor»**, men også mer beskrivende formuleringer som f.eks **«gitt to input-verdier, erstattes den største med differansen av de to, helt til en av dem er null, da returneres verdien av den andre».**

## Oppgave 22 (Største verdifall i en vektor)

I denne oppgaven returneres **4**. Den **finner det største fallet i verdi fra ett ledd til det neste**. Ved en feil var linje 2 i funksjonen d=b(2)-b(1); mens det egentlig skulle ha vært d=b(1)-b(2); Dermed ble det er lite ekstra øvingsmoment her. Det er ikke trukket for de som ikke så dette, men for de som har sett dette har jeg oversett eventuelle andre, mindre feil på denne oppgaven.

## Oppgave 23 ()

Utgår.

## Oppgave 24 (Kvasi-magisk kvadrat)

Oppgaven sjekker om hver rad og hver kolonne i en kvadratisk matrise har samme sum. Dette er en betingelse for at et kadrat er «magisk», men i tillegg skal magiske kvadrater ha ett av hvert tall fra og med 1 og til med antallet tall i kvadratet – så eksemplet som er brukt er ikke et magisk kvadrat. Funsjonen returnerer to verdier, det ene er en sannhetsverdi som forteller om matrisen har denne egenskapen og har i så fall verdien «1» (ellers verdien «0»). Den andre verdien er summen for første rad - eller summen for alle rader og kolonner dersom første parameter er sann. I denne funksjonen vil man få tilbake verdiene **«1» og «5»** som to forskjellige returverdier. En korrekt beskrivelse av funksjonen er at den **returnerer en sannhetsverdi dersom en matrise av tall er kvadratisk og har samme sum langs hver kollonne og rad, samt at den returnerer summen langs første rad.**

Det finnes forøvrig ingen magiske kvadrater i dimensjon 2×2. Dersom man faktisk skulle sjekke om det var en magisk kvadrat, måtte funksjonen utvides med f.eks med koden som er oppgitt under, men da blir oppgaven bli endel vanskeligere og inndataene blir minimum en 3×3-matrise.

```
if ~all(sort(reshape(b,1,length(b)^2))==[1:length(b)^2])
  return ;
end
```

# Oppgave 25-29 (Programmering I)

## Oppgave 25 (Gj.snittstemp og konvertering)

I denne oppgaven skulle en vektor med temperaturer i Fahrenheit regnes om til disse temperaturenes gjennomsnitt i Celcius. Omregningsformel finnes i oppgave 26 - og rekkefølgen på oppgave 25 og oppgave 26 burde vært motsatt. Det går fint å bruke funksjonen ftoc() fra oppgave 26 i denne oppgaven.

Standardløsningen er:

```
function res = avgtemp(temps)
   res = 0 ;
```

```
   for i=1:length(temps)
      res = res + ftoc(temps(i)) ;
   end
   res = res/length(temps) ;
end
```

Det er det samme om man konverterer temperaturen inne i løkka eller utenfor.

## Oppgave 26 (Konvertering fra F til C)

Det var oppgitt formel for denne konverteringen, slik at funksjonen var en ren transformasjon av denne:

```
function c = ftoc( f )
   c = (f-32)/9*5 ;
end
```

## Oppgave 27 (Minuttverdier)

Med denne deloppgaven begynner det å bli litt vanskeligere. Her skal man regne gjennomsnitt for hver linje i en matrise, men ignorere data med en bestemt verdi (-1000). Standardløsningen (som forutsetter at det er minst én gyldig verdig pr linje for å unngå deling på 0) er:

```
function res = avgprmin( temps )
   res = [] ;
   for i=1:size(temps,1)
      avg = 0 ;
      ant = 0 ;
      for j=1:size(temps,2)
         if temps(i,j) ~= -1000
            avg = avg + temps(i,j) ;
            ant = ant + 1 ;
         end
       end
       res(end+1) = ftoc(avg/ant) ;
   end
end
```

Vanskelighetsmomenter her var:

- Husk at ftoc(-1000) ikke lengre er -1000, så man må filtrere bort -1000 før konverteringe
- Man kan fjerne enkeltelementer fra en vektor, men typisk ikke fra en matrise, for da blir det 'huller' i matrisen
- Her er ikke length() nyttig, for det er ikke gitt at matrise har flere linjer enn rader eller omvendt.

Man kan også tenke seg kortere løsninger, som kanskje ikke er fullt så rett-frem og generiske.

```
function res = avgprmin( temps )
  res = [] ;
  for i=1:size(temps,1)
    res(end+1)=ftoc(mean(temps(i,find(temps(i,:)~=-1000))));
   end
end
```

## Oppgave 28 (Største temp-økning)

Her er vanskelighetsgraden økt enda litt, siden det i tillegg til mange av de samme momentene som i forrige deloppgave også kommer et krav om at siste gyldige verdi på én linje skal sammenliknes med første gyldige verdi på neste linje. Det finnes mange måter å løse dette på, men flere av dem blir raskt veldig komplekse og vil i praksis bomme på en rekke spesialtilfeller. For å komme i mål må man forsøke å finne en relativt enkel fremgangsmåte. Her skal vi vise to. Den ene itererer gjennom matrisen og holder styr på de to siste gyldige verdiene, her er en standardløsning som blir lang mest fordi det er mye nesting.

```
function res = maxtempraise( temps )
   prev = -1000 ;
   res = -1000 ;
   for i=1:size(temps,1)
      for j=1:size(temps,2)
         if temps(i,j)~=-1000
            if prev ~= -1000
               diff = ftoc(temps(i,j))-prev ;
               if diff > res
                  res = diff ;
               end
             end
          prev = ftoc(temps(i,j)) ;
          end
       end
    end
 end
```

En bedre måte å gjøre dette på er å bygge omforme matrisen til en vektor, fjerne alle verdier som er -1000, og finne største temp-økning à la vist i kodeforståelsesoppgaven.

```
function res = maxtempraise( temps )
   vect = [] ;
   for i=1:size(temps,1)
       vect = [ vect temps(i,:)] ;
   end
   vect(find(vect==-1000)) = [] ;
   for i=1:length(vect)
      vect(i) = ftoc(vect(i)) ;
```

```
   end
   res = vect(2) - vect(1) ;
   for i=3:length(vect)
       if vect(i)-vect(i-1) > res
         res = vect(i)-vect(i-1) ;
       end
    end
end
```

Det er også mulig å skrive denne endel kortere om man bruker vektorisering og litt mer spesiell funksjonalitet fra Matlab:

```
function res = maxtempraise( temps )
   vect = temps'(:) ;
   vect(find(vect==-1000)) = [] ;
   res = max(ftoc(vect(2:end))-ftoc(vect(1:end-1)))  ;
end
```

Et moment i denne oppgaven som ikke alle hadde husket på, er at ftoc() konverterer en temperatur, men ikke en temperaturdifferanse. For eksempel: 32F er 0C og 50F er 10C, men differansene mellom disse to temperaturene er +18 i F og +10 i C. Men ftoc(18) er ikke 10, det er -7,78. Derfor er det viktig å konvertere temperaturene før man regner om til temperaturdifferanse - eller ta hensyn til det på en annen måte.

## Oppgave 29 (Utskrift til fil)

Standardløsningen på dette er:

```
function writedata( filnavn, vector )
   fid = fopen( filnavn, 'a') ;
   if fid==-1
      error( ['Kunne ikke åpne filen ' filnavn] ) ;
   end
   for i=1:length(vector)-1
      fprintf(fid, '%.2f ', vector(i)) ;
   end
   fprintf(fid, '%.2f\n', vector(end)) ;
   fprintf('%d temperatures added to %s\n', ...
          length(vector), filnavn) ;
   if fclose(fid)~=0
      error(['Kunne ikke lukke filen ' filnavn] ) ;
   end
end
```

Hyppige problemer her var:

- at man fortsetter på kjøringen selv etter at det ikke var mulig å åpne fila.

- uklarhet om hvilket modus fila skal åpnes i. Med 'a' opprettes file om den ikke finnes, eller det legges til på slutten dersom den finnes. Også 'a+' vil fungere, men gir ikke noe mer i denne situasjonen. Det har ingen hensikt å forsøke å åpne fila med 'w' dersom åpning med 'a' ikke fungerte.
- I eksemplet over er det vist hvordan man får mellomrom mellom hver verdi, men ikke foran første verdi eller etter siste, kombinert med at det blir en linjeskift på slutten. Det er ikke trukket for de som har fått et ekstra mellomrom på slutten av linja.

Forsåvidt kunne man i denne oppgaven har sjekket for spesialtilfellene at lengden på vector var 0, som ville gitt en kjøretidsfeil, og dersom lengde på vector var 1, der flertallsformen «temperatures» i utskriften er litt misvisende. Det er ikke trukket for slikt under retting.

# Oppgave 30-33 (Programmering II)

## Oppgave 30 (Tre og fem)

Standardløsningen er gitt som:

```
function res = trefem( tall )
   vec = [] ;
   for i=1:tall
      if mod(i,3)==0
         vec(end+1) = 3 ;
      end
      if mod(i,5)== 0
         vec(end+1) = 5 ;
      end
   end
   res = num2str(vec) ;
end
```

Oppgaven var tvetydig formulert i hvorvidt dette skulle skrives ut eller returneres, så begge tolkninger aksepteres. Løsningen over kan modifiseres til å skrive ut ved å legge på `fprintf('%s\n',res);` som siste linje før den avsluttes. Ved å brukes num2str() unngår man også problemstillinger med innledende eller etterhengende mellomrom.

## Oppgave 31 (Innlesing fra fil)

Standardløsningen her er:

```
function res = readtext( filnavn )
   fid = fopen( filnavn ) ;
   if (fid==-1)
      error( ['Kunne ikke åpne filen ' filnavn]) ;
   end
   while ~feof(fid)
      line = fgetl( fid ) ;
```

```
      while (deblank(line))
         [res{end+1}, line] = strtok( line ) ;
      end
   end
   fclose( fid ) ;
end
```

Det er også mulig å bruke f.eks strsplit() og en rekke andre funksjoner her.

## Oppgave 32 (Ordfrekvenser)

Standardløsningen på dette er:

```
function [words, freqs] = countfreqs ( input )
   words = {} ;
   for i=1:length(input)
      found = false ;
      for j=1:length(words)
         if strcmp(words{j},input{i})
            freqs(j) = freqs(j) + 1 ;
            found = true ;
         end
      end
      if (~found)
         words(end+1) = input(i) ;
         freqs(length(words)) = 1 ;
      end
   end
end
```

Dette er forsåvidt en temmelig brute-force-løsning, da den må gå igjennom lista med ord som allerede er funnet for hvert nytt ord som leses inn. Dette vil bare være et problem med inndata som er svært store og varierte. Dersom man ønsker å effektivisere, ville det være mulig å sortere words, slik at man kan binærsøke seg frem til rett sted - men dette er ikke tatt hensyn til under sensur.

## Oppgave 33 (Ordlengder)

I denne oppgaven kom en ordliste inn som parameter som et cell-array, og det skulle returneres en vektor av heltall, der posisjonen i vektoren angav hvor mange ord av denne lengden det fantes i ordliste som var parameter. Standardløsningen er:

```
function res = wordlength( words )
   maxlen = 0 ;
   for i=1:length(words)
      if length(words{i}) > maxlen
         maxlen = length(words{i}) ;
      end
```

```
   end

   res = zeros(1,maxlen) ;
   for i=1:length(words)
      res(length(words{i})) = res(length(words{i})) + 1 ;
   end
end
```

Utfordrende elementer her var

- syntaksen word{i} siden det er et cell array
- bruk av variabelnavnet length sammen med den innebygde funksjonen length()
- sikre at man ikke inkrementerer et udefinert element i vektoren
- huske at zeros(rad,kol) skal ha to parametre, for zeros(rad) gir en rad×rad matrise.

Flere har initialisert returverdien til f.eks zeros(1,20), med en antakelse om at det ikke er mer enn 20 tegn i noe ord. Det er en dårligere løsning enn den som er gitt over, siden den er mindre generell. Det er også en feil løsning, med mindre man sikrer at man ikke forsøker å prosessere ord som er mer enn 20 tegn lange.
En kortversjon med én løkke vil være:

```
function res = wordlength( words )
   res = [] ;
   for i=1:length(words)
      len = length(words{i}) ;
      if length(res)<len
         res(len) = 1 ;
      else
         res(len) = res(len) + 1 ;
      end
   end
end
```

Denne versjonen utnytter at om man setter en verdi et stykke uti en vektor, så «fylles» det på med det nødvendige antallet nuller i mellom.
En rekke kandidater hadde kodet den siste løkka i standardløsningen som:

```
   for i=1:maxlen
      for j=1:length(words)
         if i==length(words{j})
            res(i) = res(i) + 1 ;
         end
      end
   end
```

Men her kan man eliminere både den ytre for-løkka og if setningen, og sitte igjen med:

```
   for i=1:length(words)
      res(length(words{i})) = res(length(words{i})) + 1 ;
   end
```

En annen artig løsning er denne, hvor man først lager array over hvert ords lengde, noe som gjør det lett å finne lengste ord og antall ord av en bestemt lengde. Med nedtelling er det helt unødvendig å preallokere plass siden alle lavere elementer initialiseres til 0.

```
function res = wordlength( words )
   for i=1:length(words)
      tmp(i) = length(words{i}) ;
   end
   for j=max(tmp):-1:1 ;
      res(j) = sum(tmp==j) ;
   end
end
```

## Karakterberegning

Eksamen inneholdt 170 poeng, men to oppgaver på 10 og 2 poeng er trukket, og det er tatt hensyn til i utregningen. Det er også tatt hensyn til effekten av at man vil få rett på en fjerdedel av spørsmålene på multiple choice ved vill tipping, og karakterskalaen er justert for å ta hensyn til dette.

Det er tatt utgangspunkt i det anbefalte grenseverdiene for karakterer [100 89 77 65 53 41 0] og det er skalert slik at karaktergrensene ble:  (A) >= 149 > (B) >= 129 > (C) >= 109 > (D) >=  89 > (E) >= 68 > (F)