NTNU
Kunnskap for en bedre verden

Institutt for datateknikk og informasjonsvitenskap

Eksamensoppgave i TDT4105 Informasjonsteknologi – grunnkurs, med Matlab – LØSNINGSFORSLAG

Løsningsforslag for følgende oppgaver:

- Oppgave 1: Flervalgsoppgave (25%)
- Oppgave 2: Kodeforståelse (15%)
- Oppgave 3: Programmering reisetid (20%)
- Oppgave 4: Programmering sensur (40%)
- Svarark for hurtigsensur

Oppgave 1: Flervalgsoppgave (25%)

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
b	b	c	d	b	a	c	a	b	c	b	d	a	c	b	a	a	d	a	a

Oppgave 2 Kodeforståelse (15%)

Oppgave 2a (5%)
Hva blir skrevet ut til skjerm når du kjører koden som vist under? (3 %)

```
JULENISSEN
```

Forklar med en setning hva funksjonen **`mystery`** gjør (2 %)

Funksjonen plukker ut annenhver bokstav fra de to listene og starter med liste B.

Oppgave 2b (5%)
Hva blir skrevet ut til skjerm når du kjører koden som vist under? (3 %)

```
64
```

Forklar med en setning hva funksjonen **`compute`** gjør (2 %)

Rekursiv funksjon som multipliserer et tall med 2 ganger tallet så lenge tallet er mindre enn 10, 1*2*4*8 = 64.

Oppgave 2c (5%)
Hva blir skrevet ut til skjerm når du kjører koden som vist under? (3 %)

[8, 7, 6, 5, 3, 2, 1]

Forklar med en setning hva funksjonen **a** gjør (2 %)

Sorterer en liste med tall i synkende rekkefølge («selection sort med bytting»).

Oppgave 3 Programmering reisetid (20%)

Oppgave 3a (5%)
Lag funksjonen **readTime** …

```
function res = readTime()
h = input('Enter hour: ') ;
while h<0 || h>23 || h~=fix(h)
    fprintf('- ERROR: Hour must be between 0 and 23!\n');
    h = input('Enter hour: ') ;
end
m = input('Enter minute: ') ;
while m<0 || m>59 || m~=fix(m)
    fprintf('- ERROR: Minute must be between 0 and 59!\n');
    m = input('Enter minute: ') ;
end
s = input('Enter second: ') ;
while s<0 || s>59 || s~=fix(s)
    fprintf('- ERROR: Second must be between 0 and 59!\n');
    s = input('Enter second: ') ;
end
 res = [ h, m, s ] ;
end
```

Oppgave 3b (5%)
Lag funksjonen **convertTime …**

```
function ret = convertTime(verdi, opt)
if strcmp(opt, 'time')
    h = floor(verdi/3600) ;
    m = floor((verdi-h*3600)/60) ;
    s = verdi - h*3600 - m * 60 ;
    ret = [ h, m, s] ;
elseif strcmp(opt, 'sec')
    ret = (verdi(1)*60 + verdi(2))*60 + verdi(3) ;
else
    error('incorrect mode') ; %Extra check
end
end
```

Oppgave 3c (5%)
Lag funksjonen **travelTime** …

```
function travelTime()
fprintf('Give departure time in hour, minute and second:\n');
startTime = convertTime(readTime(), 'sec') ;
fprintf('Give arrival time in hour, minute and second:\n');
endTime = convertTime(readTime(), 'sec') ;
while (endTime < startTime)
    fprintf('- ERROR: Arrival time must be later than Departure time\n') ;
    fprintf('Give arrival time in hour, minute and second:\n');
    endTime = convertTime(readTime(), 'sec') ;
end
travTime = convertTime(endTime-startTime, 'time');
fprintf('Traveltime: %d hr, %d min, %d sec\n', ...
     travTime(1), travTime(2), travTime(3));
end
```

Oppgave 3d (5%)
Lag funksjonen **analyzeBusRoutes** …

```
function analyzeBusRoutes( busses )
korteste.tid = 24*60*60 ;
lengste.tid = 0 ;
for i=1:size(busses,1)
    diff = (busses(i,4)-busses(i,2))*60 + busses(i,5)-busses(i,3);
    if diff>lengste.tid
        lengste.tid = diff ;
        lengste.nr = busses(i,1) ;
    end
    if diff<korteste.tid
        korteste.tid = diff ;
        korteste.nr = busses(i,1) ;
    end
end
tid = convertTime(lengste.tid*60, 'time') ;
fprintf('The slowest bus route is bus nr. %d and it takes %d hour, %d min.\n', ...
    lengste.nr, tid(1), tid(2)) ;
tid = convertTime(korteste.tid*60, 'time') ;
fprintf('The fastest bus route is bus nr. %d and it takes %d hour, %d min.\n', ...
    korteste.nr, tid(1), tid(2)) ;
end
```

Oppgave 4 Programmering Sensur (40%)

Oppgave 4 a) (5%)
Lag starten på hovedprogrammet (ikke funksjon)…

```
NTNU_scores = [89 77 65 53 41 0]
NTNU_letters = {'A', 'B', 'C', 'D', 'E', 'F'}

TASKS = {'1', '2a','2b','2c', '3a','3b','3c','3d', ...
    '4a','4b','4c','4d','4e','4f','4g','4h'}

WEIGHTS = [25, 5,5,5, 5,5,5,5, 5,5,5,5,5,5,5,5]
```

Oppgave 4 b) (5%)
Lag funksjonen **`makeArray`** …

```
function array = makeArray( numbers, texts )
%   Make a data-structure (cell array) from numbers and characters
array = cell( length(numbers), 2 );
for i=1:length(numbers)
    array{i,1} = numbers(i);
    array{i,2} = texts{i};
end %for
end
```

Oppgave 4 c) (5%)
Lag funksjonen **computeScore** …

```
function total = computeScore( points, weight )
    total = 0;
    for i=1:length( points )
        total = total + points(i)/10 * weight(i);
    end %for

    %Alternative: Vectorized
    total = sum( points/10 .* weight );
end %function
```

Oppgave 4 d) (5%)
Skriv en funksjon **score2Letter** …

```
function letter = score2letter( score, limitLetters )
%letter=''; %Don't return anything if the input is not >0
%assuming limitLetters is sorted in descending score order:
for i=1:length(limitLetters)
    if score >= limitLetters{i,1}
        letter = limitLetters{i,2};
        return; %or break
    end
end
end %function
```

Oppgave 4 e) (5%)
Skriv en funksjon **addCandidate** …

```
function addCandidate( candidateNr, scores, weight )
fid = fopen('eksamen.txt','a');
if fid<0
    error('Could not open the file eksamen.txt\n');
end

text = sprintf('%d\t', candidateNr);
for i=1:length(scores)
    text = sprintf('%s\t%d ', text, scores(i));
end
fprintf( fid, '%s\t%.1f\n', text, computeScore( scores, weight ) );
fclose(fid);
end %function
```

Oppgave 4 f) (5%)

Skriv en funksjon **readResultFile …**

```
function res = readResultFile(filename)
%Convert a file-table of strings to a table of int
fid = fopen(filename,'r');
if fid==-1
    error( Could not open the file' ) ;
end
res = [] ;
while ~feof(fid)
    line = fgetl(fid);
    res(end+1,:) = str2num(line) ;
end
fclose(fid) ;
end
```

Oppgave 4 g) (5%)

Skriv en funksjon **checkResultOK** …

```
function ok = checkResultOK( filename, weight )
ok = true; %Antar at alt er ok
data = load( filename ); %or readResultFile( filename )
[rows,cols] = size( data );
for i = 1:rows
    knr = data(i,1);
    scores = data( i, 2:cols-1 );
    total = data( i, cols );
    %sjekk at ingen kommer mer enn en gang
    if ( sum( data(1:i,1) == knr ) > 1 )
        fprintf( 'ERROR: Candidate %d appears more than once!\n', knr );
        ok = false;
    end %Dobbelt-linje
    %sjekk at ingen har fått <0 eller >10 poeng
    if sum ( scores <0 ) + sum( scores >10 ) > 0
        fprintf('ERROR: Candidate %d scores are not between 0-10!\n', knr);
        ok = false;
    end
    %sjekk at totalen er korrekt
    if ( total ~= computeScore( scores, weight ) )
        fprintf('ERROR: Candidate %d has wrong total score!\n', knr);
        ok = false;
    end
end
```

Oppgave 4 h) (5%)

Skriv en funksjon **listAll** …

```
function rows = listAll( filename, limitLetters )
data = load( filename ); %or readResultFile( filename )
[rows, cols] = size( data );
data = sortrows( data, cols );
for i=1:rows
    fprintf( '%5i %5.1f %c\n', data(i,1), data(i,cols), ...
        score2letter( data(i,cols), limitLetters ) );
end %for
end %function
```

Sensor-veiledning

- 10 poeng: Helt riktig løsning. Kan være en annen løsning enn fasit.
- 7-9 poeng: Små syntaktiske feil.
- 5-7 poeng: Mindre semantiske og syntaktiske feil.
- 3-5 poeng: Noe riktig tankegang, men en del semantiske/syntaktiske feil.
- 2 poeng: Korrekt funksjonshode og returnering av resultat.
- 1 poeng: Noe riktig. For eksempel funksjonshode eller returnering av resultat.

A>89, B>77, C>65, D>53, E>41, F<41

Transparent for flervalgsoppgave: Viser riktige svar (dekker over feil svar)

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				

Transparent for flervalgsoppgave: Viser feil svar (dekker over riktig svar)

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				