i

# Forside

Department of Computer Science

Examination paper for TDT4105 IT-Grunnkurs (Matlab)

Academic contact during examination: Anders Christensen
Phone: 91897181
Examination date: 3. december 2018
Examination time (from-to): 09:00 13:00
Permitted examination support material: D
Other information:

Students will find the examination results in Studentweb. Please contact the department if you have questions about your results. The Examinations Office will not be able to answer this.

i

# Introduction

Every question in the multiple choice part gives 2 points if answerd correctly, and zero points of answered incorrectly.
It is possible to get up to 20x2 points on the multiple choice part, which counts as 40 points in the exam, which can have a total of 170 points – or 23,5% of the whole exam.

If you are uncertain which of two alternatives that is correct, select the alternative that is most generally correct.

1

# Hex to des

What the hexadecimal number FAB in the decimal number system?

**Select one alternative:**

○ 4011

○ -2034

○ 256

○ It cannot be represented

Maximum marks: 2

## 2 Conversion

What is used to convert analogue sound or visual data to digital format?

**Select one alternative:**

○ IEEE-format

○ Fitting of curves

○ A digital to analogue converter (DAC)

○ Sampling

Maximum marks: 2

## 3 Oslo

Which hexadecimal text string represents the text string ‘OSLO’ in ASCII?

**Select one alternative:**

○ F4OSLOF4

○ 30353139

○ 64686A7B

○ 6D726B6D

Maximum marks: 2

## 4 Nyquist

What do Nyquist’s theorem tell us something about?

**Select one alternative:**

○ Minimum delay between different sound channels in order to prevent interference.

○ The required sampling rate in relation to the highest frequency that must be reproduced

○ The required amount of check-sum data to ensure correct recording of sound

○ The balance point of the volume (i.e. in amplitude) between loud and weak sounds

Maximum marks: 2

## 5 RAM

What does ‘random’ i RAM signify?

**Select one alternative:**

○ That it is random where in memory information is stored

○ That the computer retrieves information from a random location in memory

○ That all elements can be retrieved directly

○ That it is random what is stored in memory

Maximum marks: 2

## 6 Vacuum tubes

What is used today and has replaced Vacuum tubes?

**Select one alternative:**

○ Assembly

○ CPU

○ RAM

○ Transistors

Maximum marks: 2

## 7 SSD

What does SSD mean?

**Select one alternative:**

○ Sequential state drive

○ Simple software development

○ Simple system development

○ Solid state drive

Maximum marks: 2

## 8 Silicon

Why is silicon use so extensively in electroncs?

**Select one alternative:**

○ It is a cheap and environmentally friendly conductor

○ It is a semiconductor

○ There are huge resources of silicon

○ It is a efficient isolator for komponents

Maximum marks: 2

## 9 ALU

What is the main task for ALU?

**Select one alternative:**

○ Point to the correct memory address

○ Retrieve data from memory

○ Initiate the operating system

○ Execute calculations

Maximum marks: 2

## 10 Assembler language

What is an assembly language?

**Select one alternative:**

○ A programming language that resembles Python

○ It is another word for pseudo code, i.e a compact and informative description of an algoritm

○ An alternative form for machine language that uses letters and normal numbers that humans can understand

○ The first programming language that was ever used

Maximum marks: 2

## 11 Pipelining

What is pipelining

**Select one alternative:**

○ A technique where one sends data between the different parts of the computer, in «pipes»

○ A technique where a CPU can execute several instructions in parallel

○ An expression for what happends when one writes a lot of data to the hard disk at once.

○ A technique that provides a secure tunel between your computer and a server

Maximum marks: 2

## 12 Location

I a modern computer memory, every location consists of ...

**Select one alternative:**

○ 1 gigabyte

○ 1 byte

○ 10 bytes

○ 1 bit

Maximum marks: 2

## 13 WAN

What does the abbreviation WAN means in relation to networks?

**Select one alternative:**

○ Weak Area Network

○ World Area Network

○ Wired Area Network

○ Wide Area Network

Maximum marks: 2

## 14 IPv4-adresses

How many bits are there in an IPv4 address

**Select one alternative:**

○ 128

○ 64

○ 32

○ 16

Maximum marks: 2

## 15 Reliability

Which of the following standards ensures relieable transmission of packages over the Internet?

**Select one alternative:**

○ TCP

○ IP

○ Intranet

○ UDP

Maximum marks: 2

## 16 Duplicates

Which method is used to handle duplicates and situations where packages arrives in incorrect order?

**Select one alternative:**

○ Flow control

○ Sequencing

○ ACK (retransmisjon)

○ Replay

Maximum marks: 2

## 17 IoT

What does IoT mean?

**Select one alternative:**

- ○ Internet over Time
- ○ Introduction over Technology
- ○ Internet of Things
- ○ Internet of Technology

Maximum marks: 2

## 18 Fire walls

What is NOT true about firewalls?

**Select one alternative:**

- ○ A security technology for data integrity
- ○ They supervise and control traffic in and out of a network
- ○ They protects against problems from the outside
- ○ They are usually built into switches and routers

Maximum marks: 2

## 19 Digital signature

What does it mean to add a digital signature to a message?

**Select one alternative:**

- ○ To sign a contracts over the network
- ○ A message that is sent digitally
- ○ That a private key is used to encrypt a message
- ○ A message where the data is divived into mange packages and sent digitally

Maximum marks: 2

20

## Data integrity

Which security technology underpins data integrity

**Select one alternative:**

○ Password

○ Digitale signatures

○ Encryption

○ Hashing

Maximum marks: 2

i

## Introduction

In this section of exercises two tasks are to be answered for each question.

Firstly, you shall write a short description of wthat the listed function does, as you interpret it. Try to describe its effects rather than how it does this. The shorter, the better.

The other thing you shall do, is to write the value that will be returned from the function when it is called with the given parameter.

Every exercise counts up to 10 points, or a total maxium of 40 points of a total maxium of 170 points for the whole exam. Thus this section counts for 23,5% of the whole exam.

## 21 Kodeforståelse 1

Describe briefly what the following function does.
What is the return value if it called as `myst1(14,42)`?

```
function a = myst1( b, c )
  while b && c
    if b>c
      b = b - c ;
    else
      c = c - b ;
    end
  end
  a = b + c ;
end
```

**Fill in your answer here**

Format

Words: 0

Maximum marks: 10

## 22 Kodeforståelse 2

Describe briefly what the following function does.
What is the return value from this function if it is called as `myst2(vect)`, and the vector vect has the value `[ 1 2 5 1 9 7 4 5 2 9 6 ]` ?

```
function d = myst2(b)
   d = b(2) - b(1) ;
   for c=2:length(b)-1
      if b(c) - b(c+1) > d
         d = b(c) - b(c+1) ;
      end
   end
end
```

**Fill in your answer here**

Format

Words: 0

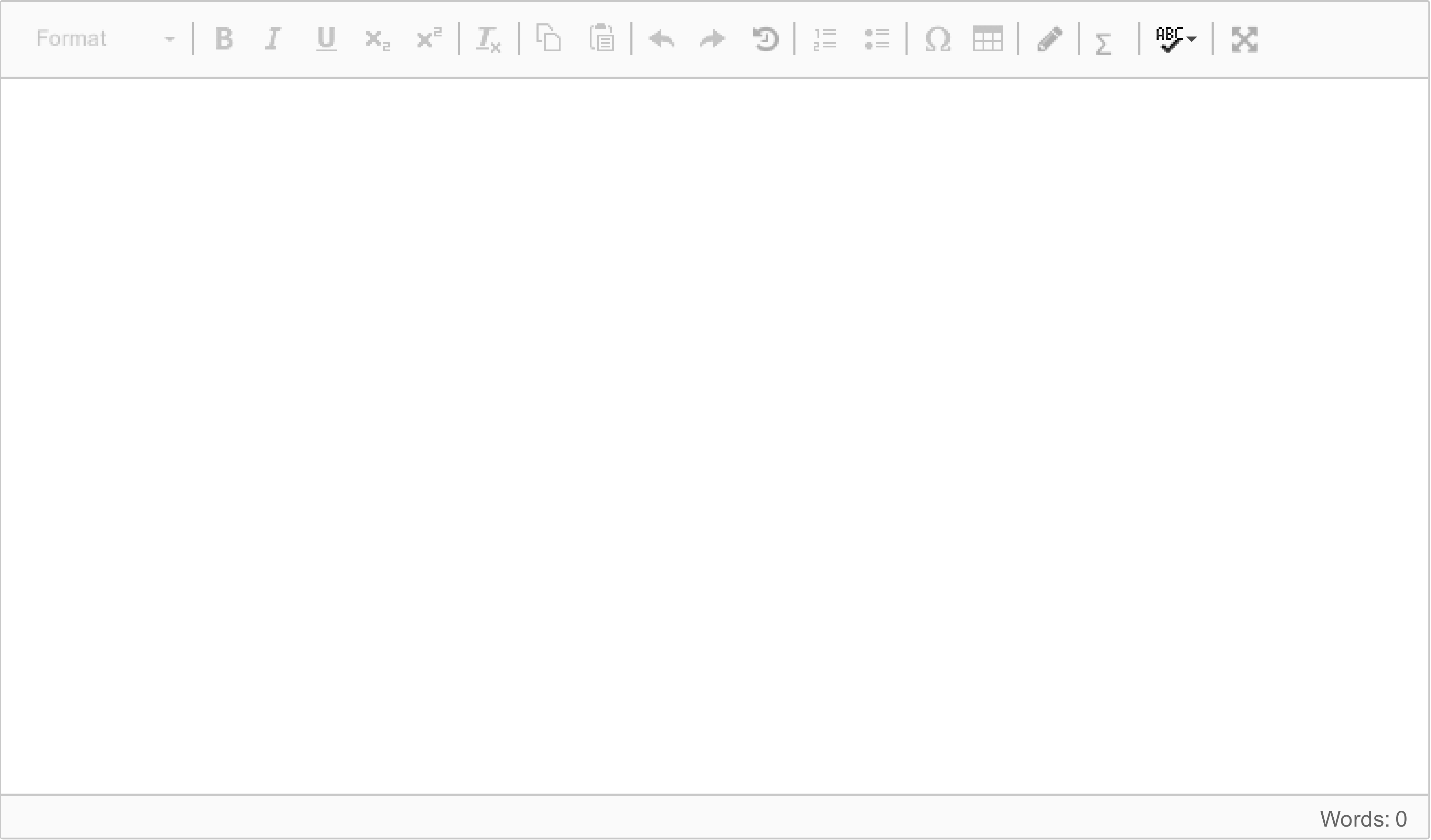
Maximum marks: 10

## 23 Kodeforståelse 3

Describe briefly what the following function does.
What is the return value from this function if it is called as `myst3(myst3(b))`, and the valiable b has the value `'testing'`.

```
function d = myst3(b)
  d = '' ;
  for c=1:length(b)
    e = b(c) - 'a' + 13 ;
    if e<1
      e = e + 26 ;
    end
    d(e) = char(e + 'a') ;
  end
end
```

**Fill in your answer here**



Maximum marks: 10

## 24 Kodeforståelse 4

Describe briefly what the following function does.
What is the return value from this function if it is falled as `myst4(a)`, and
the variable a has the value `[1 4; 4 1]`?

```
function [ a, c ] = myst4(b)
  c = sum(b(1,:)) ;
  a = false ;
  if size(b,1) ~= size(b,2)
    return
  end

  for d=1:size(b,1)
    if c~=sum(b(d,:)) || c ~= sum(b(:,d))
      return
    end
  end
  a = true ;
end
```

**Fill in your answer here**

Format

Words: 0

Maximum marks: 10

## i Introduction

In this section, you what write function for the operation of a laboratory setup. Throughout the exercise, simple and fixex input data is used. However, the code that you write must be able to handle general data of arbitrary length.

Every exercise in this section count 10 points, so that this programming exercise counts a total of 50 points of the total maximum of 170 points for the whole exam. Thus it counts 29,5% of the whole exam.

## 25 Gjennomsnitt

Write a function that takes a vector with temperature measurements in Fahrenheit. It shall return the average temperature in Celcius. You can not use the built-in Matlab functions sum() or mean() in this exercise. The function that you shall write shall be named avgtemp(). Below is an example of how it can be used. Remember that input is in Fahrenheit and output is in Celcius.

```
>> vekt = [ 53 54 59 56 59 67 66 58 ] ;
>> avgtemp( vekt )
ans =
    15
```

**Fill in your answer here**

Format

Words: 0

Maximum marks: 10

## 26 Konvertering

Initially, you shall write a function ftoc() that converts a temperature from degrees Fahrenheit to degrees Celcius. The function shall take a parameter that is the number of degrees in Fahrenheit. The relationship between the two is that you get degrees Celcius is given by the formula (f-32)*5/9 where f is degrees Fahrenheit.

Usage example:

```
>> c = ftoc( 56 )
c =
    13.3333
```

**Fill in your answer here**

Format

Words: 0

Maximum marks: 10

# 27 Minuttverdier

A temperature logger returns temperature readings every 11th second, and stores up to 6 temperatures per minute, using the the Fahrenheit scale. The data is stored in a table, where each line contains data for a single minute.

Since Matlab cannot have undefined single values in a table, we will use the temperature -1000 to signify that this position in the table contains an undefined temperature, and that it can be ignored. For instance, there might be less that six temperatures for some minutes, and it might happen that some temperature readings fail.

The function shall calculate the average for the valid temperatures measurements (i.e. the values other than -1000) that is defined for each minute (i.e. for each line in the table).

You may assume that there is at least one valid temperature measurement for each minute (i.e. for each line).

Write a function avgpermin() that takes such a table as input parameter, and that returns a vector with temperatures in Celcius for the avergate for each minute. Remember that the input data is in Fahrenheit and that the output data shall be in celcius.

```
>> temps = [ 54 56 64 62 -1000 -1000; ...
    66 68 -1000 62 58 -1000; ...
    58 59 62 63 63 63 ] ;
>> avgpermin( temps )
ans =
     15.0000   17.5000   16.2963
```

**Fill in your answer here**

Format

Words: 0

Maximum marks: 10

## 28 Største temp-økning

Write a function maxtempraise() to find the greatest raise of temperature between two immediately following temperatures in the table. A temperature immediately follows another, even if there are invalid values (i.e. the value -1000) between them in the input table. The last valid value in one line is assumed to be followed by the first valid value in the next line. Remember that indata is in Fahrenheit and that output shall be retuned in Celcius.

```
>> temps = [ 54 56 64 62 -1000 -1000; ...
   66 68 -1000 62 58 -1000; ...
   58 59 62 63 63 63 ] ;
>> maxtempraise( temps )
ans =
    4.4444
```

**Fill in your answer here**

Format

Words: 0

Maximum marks: 10

## 29 Skriv ut til fil

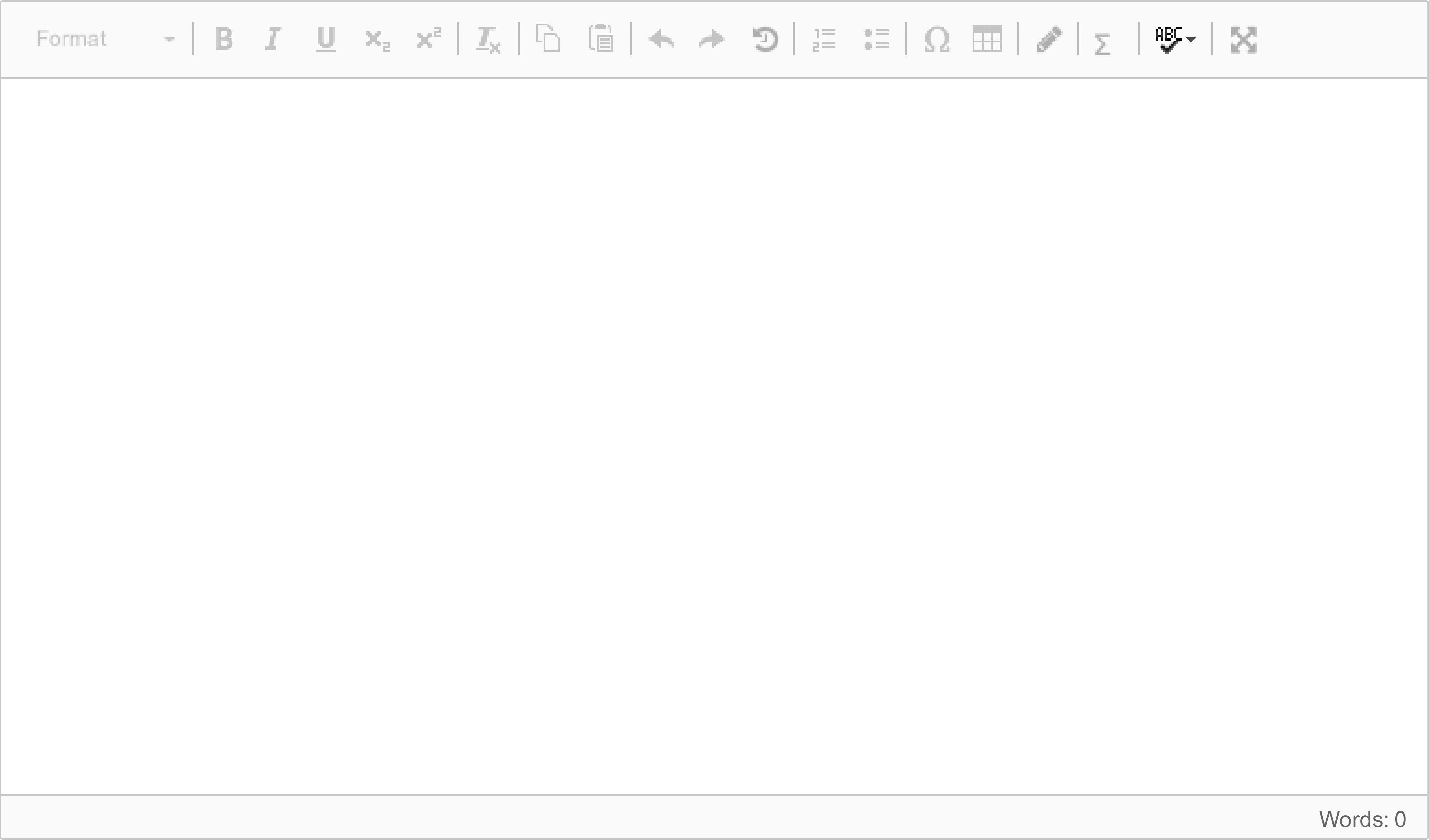
Write a function called writedata() that writes the data for the previous exercise to a file. The file name shall be given as the first parameter, while the vector containing the data shall be given as the second parameter. If the file exist, data shall be added to the end of it. If the file does not exist, it shall be created.

The data shall be written as text data, and the numbers shall be written with two decimals precision and one blank space between numbers.

You must check taht the file was correctly opened, and that it was correctly closed. If there is any error during opening or closing, a short error message shall be written. If the function succeed in writing the data, it shall write a brief status message: **`<N> temperatures added to <filename>`**

```
>> temps = [ 54 56 64 62 -1000 -1000; ...
   66 68 -1000 62 58 -1000; ...
   58 59 62 63 63 63 ] ;
>> vector = avgpermin( temps )
ans =
    59.0000   63.5000    61.3333
>> writedata( 'logdata.out', vector ) ;
3 temperatures added to logdata.out
>>
```

**Fill in your answer here**



Maximum marks: 10

## i Nyttige funksjoner i Matlab

### Useful Matlab functions and commands

**blanks** - String of blanks. *blanks(n)* is a string of n blanks. Use with *disp()*, e.g. *disp( [ 'xxx' blanks(20) 'yyy' ] )*

**cell2mat** - Converts a cell array into an ordinary array. The elements of the cell array must all contain the same data type, and the resulting array is of that data type.

**fix** - Round towards zero. *fix(x)* rounds the elements of *x* to the nearest integers towards zero.

**fclose** - Close file. *st = fclose(fid)* closes the file associated with file identifier *fid*, which is an integer value obtained from an earlier call to *fopen()*. *fclose()* returns 0 if successful or -1 if not.

**feof** - Test for end-of-file. *st = feof(fid)* returns 1 if the end-of-file indicator for the file with file identifier *fid* has been set, and 0 otherwise. The end-of-file indicator is set when a read operation on the file associated with the *fid* attempts to read past the end of the file.

**fgetl** - read line from file, discard newline character. *tline = fgetl(fid)* returns the next line of a file associated with file identifier *fid* as a MATLAB string. The line terminator is NOT included. Use *fget*s() to get the next line with the line terminator INCLUDED. If just an end-of-file is encountered, -1 is returned.

**find** - Returns the linear indexes of non-zero elements in a matrix. *find([0 1 0 1 0])* returns [2 4]. If the first parameter has more than one row, a column vector containing the linear indexes of non-zero elements are returned. An optional second parameter set the maximum number of indexes to return.

**fopen** - Open file. *fid = fopen(filename,permission)* opens the file *filename* in the mode specified by PERMISSION:
'r' - open file for reading
'w' - open file for writing; discard existing contents
'a' - open or create file for writing; append data to end of file
'r+' - open (do not create) file for reading and writing
'w+' - open or create file for reading and writing; discard existing contents
'a+' - open or create file for reading and writing; append data to end of file

**fprintf** - Write formatted data to file. *count = fprintf(fid,format,A,...)* formats the data in the real part of array A (and in any additional array arguments), under control of the specified *format* string, and writes it to the file associated with file identifier *fid*. *count* is the number of bytes successfully written. *fid* is an integer file identifier obtained from *fopen*(). It can also be 1 for standard output (the screen) or 2 for standard error. If *fid* is omitted, output goes to the screen. *format* is a string containing ordinary characters and/or C language conversion specifications. Conversion specifications involve the character %, optional flags, optional width and precision fields, optional subtype
specifier, and conversion characters d, i, o, u, x, X, f, e, E, g, G, c, and s. The special formats \n, \r, \t, \b, \f can be used to produce linefeed, carriage return, tab, backspace, and formfeed characters respectively. Use \\ to produce a backslash character and %% to produce the percent character.

**global** - Define global variable. *global X Y Z* defines *X*, *Y*, and *Z* as global in scope (scope can be functions/programs).

**input** - Read a value from the keyboard and into a variable. *answer=input(str)* prints str as a prompt, reads a number and assigns it to *answer*. If character string is to be read, use the optional second parameter ‘s’.

**isempty** - Determine whether array is empty This MATLAB function returns logical 1 (true) if A is an empty array and logical 0 (false) otherwise. *TF = isempty(A)*

**length** - The length of vector. *length(X)* returns the length of vector X. It is equivalent to *max(size(X))* for non-empty arrays and 0 for empty ones. use size(X,1) and size(X,2) etc to get the size in specific dimensions.

**load** - Loads data from filename. *load(filename)* loads data from filename. If filename is a MAT-file, then *load(filename)* loads variables in the MAT-File into the MATLAB® workspace. If filename is an ASCII file, then *load(filename)* creates a double-precision array containing data from the file.

**max** - finds the highest element in a vector, or the highest element in each column of a matrix.

**min** - finds the lowest element in a vector, or the lowest element in each column of a matrix.

**mod** - Modulus after division. *mod(x,y)* is x - n.*y where *n = floor(x./y)* if y ~= 0.

**num2str** - Convert numbers to a string.

**randi** - Pseudorandom integers from a uniform discrete distribution. *R = randi(IMAX,N)* returns an N-by-N matrix containing pseudorandom integer values drawn from the discrete uniform distribution on *1:IMAX*. Further: *randi(IMAX,M,N)* or *randi(IMAX,[M,N])* returns an M-by-N matrix.

**rem** - Remainder after division. *rem(x,y)* is *x - n.*y* where *n = fix(x./y)* if y ~= 0.

**repmat** - returns a matrix constructed from concatenating its first parameter multiple times. For instance *repmat(foo, 3, 4)* replicates *foo* three times vertically and four times horisontally. The function *repmat('ab',1,3)* returns 'ababab'.

**round** - Rounds to nearest decimal or integer. *Y = round(X)* rounds each element of *X* to the nearest integer. If an element is exactly between two integers, the round function rounds away from zero to the integer with larger magnitude. *Y = round(X,N)* rounds to N digits

**size** - The size of array. *D = size(X)*, for M-by-N matrix X, returns the two-element row vector. D = [M,N] containing the number of rows and columns in the matrix. Further, *size(X,D)* gives the size of *X* in dimention *D*, where 1 means number of rows and 2 means number of columns.

**sortrows** - Sort array rows. This MATLAB function sorts the rows of a matrix, table or cell arry in ascending order, based on values in first column: *B = sortrows(A)*. More generally: *B = sortrows(A [, column [, order]])*. Here, *column* can be a vector of integers, where the second (third etc) element gives the secondary (tertiary etc) column that determine the sort order for elements that have the same value in the primary sort colum. If *order* is present, it must be either 'ascending' or 'descending' to specify the sorting order.

**sprintf** - Basically the same as *fprintf()* without the first, optional parameter fid, but instead of printing to the standard output, it returns the constructed string value.

**sscanf** - Extracts values from a string according to a format string. Opposite of *fprintf()*. *A=sscanf(’12/11-2014’,’%d/%d-%d’)* returns a column vector containing the values 12, 11, and 2014.

**strcmp** - Compare strings. *TF = strcmp(S1, S2)* compares the strings *S1* and *S2* and returns logical 1 (true) if they are identical, and returns logical 0 (false) otherwise.

**strsplit** - Splits the first (string) parameter into a cell array of substrings, according to the delimiter string given as the second parameter. *strsplit('one, two, three', ', ')* results in {'one', 'two', 'three'}. Multiple alternative delimiters can be specified using a cell array as the second parameter.

**strtok** - separates the first token of a string from the rest of that string.
*[token, rest] =strtok (‘ first second’, delim)* sets *token* to ‘first’ and *rest* to ‘ second’. The optional parameter *delim* contains a list of delimiter characters – where the space character is default. Any delimiter characters before the first token are ignored, but delimitors between first token and the rest of the string is preserved at the front of *rest*.

**str2num** - Convert string matrix to numeric array. *X = str2num(S)* converts a character array representation of a matrix of numbers to a numeric matrix. For example, if *S=['12'; '34']* then *str2num(S)* => [ 12; 34 ]. Further, if *S=’abc’* then *str2num(S)=> []*

**sum** - The sum of elements. *S = sum(X)* is the sum of the elements of the vector *X*. If *X* is a matrix, *S* is a row vector with the sum over each column.

# Introduction

Write functions that answers each exercise in this section.

Every exercise in this section counts 10 points, and this section counts a maxium of 40 points out of a maximum of 170 for the whole exam. Thus this exercise counts for 23,5% of the whole exam.

## 30 Three and five

Write a function that is called trefem(). This shall take one paramter, which will be a positive integer. The function that interate though all integers from and including 1 until and including the value of the parameter. If the number is divisable by 3, it shall write out the text ‘3’. If the numer of divisable by 5, it shall write out the tekst ‘5’. If the number is divisable by both 3 and 5, it shall first write out ‘3’ and then write ‘5’. There shall be one blank space between each time is writes something out.

```
>> trefem( 13 )
ans =
    '3 5 3 3 5 3'
>> trefem( 33 )
ans =
    '3 5 3 3 5 3 3 5 3 5 3 3 5 3 3 5 3'
```

**Fill in your answer here**

Format

Words: 0

Maximum marks: 10

## 31 Read in from file

Write a function called readtext() which takes a file name as parameter and reads the named file as a text file. Alle words in the text shall be isolated, and it shall return a cell array with all the words. The content of the first cell in this cell array is the first word in the text. The content in the second cell is the second word in the tekst, etc. If we assume that the first line of the text file is: **'This is the first line'**, then you will get:

```
>> carray = readtext( 'textfile.txt' ) ;
>> carray{1}
ans =
    'This'
>> carray{2}
ans =
    'is'
```

**Fill in your answer here**

Format

Words: 0

Maximum marks: 10

## 32 Word frequencies

Write a function countfreqs(carray) that takes a cell array of the type that was returned from the previous exercise. The function shall find all unique words in the input parameter, and they shall be returned as the first return value, as a cell array that contains this unique words. This list does not need to be sorted in any way, but all words shall be include, but no words shall be included twice.

The other – second – return value shall be a vector of integers. There shall be as many integers in this vector as there are words in the first return value. Every integer specify the number of time the corresponding word occurs in the input parameter. If we assume that the word ‘first’ occurs only once, while the word ‘line’ occurs four times, we could get the following:

```
>> [ words, freqs ] = countfreqs( carray ) ;
>> words{1}
ans =
    'first'
>> freqs(1)
ans =
    1
>> words{2}
ans =
    'line'
>> freqs(2)
ans =
    4
```

**Fill in your answer here**

Format

Words: 0

Maximum marks: 10

## 33 Word length

You shall now make a list of the distribution of length of words in a list of word. Write a function wordlength() that takes as the first parameter a list of words that was returned from readtext() from an earlier exercise. The function shall return a vector of integers, where the first element specifies the number of words of length one character, the second element specifies the number of words of length two characters, etc. Word lengths that does not occur in the indata shall be represented as zero in the return value. If we assume that the indata contain three words of one letter, eight words of two letter and 12 words of three letters, we will get:

```
>> words = readtext( 'textfile.txt' ) ;
>> lengthlist = wordlength( words ) ;
>> lengthlist
lengthlist =
    3    8   12
```

**Fill in your answer here**

Format

Words: 0

Maximum marks: 10

## Kopi av Nyttige funksjoner i Matlab

### Useful Matlab functions and commands

**blanks** - String of blanks. *blanks(n)* is a string of n blanks. Use with *disp()*, e.g. *disp( [ 'xxx' blanks(20) 'yyy' ] )*

**cell2mat** - Converts a cell array into an ordinary array. The elements of the cell array must all contain the same data type, and the resulting array is of that data type.

**fix** - Round towards zero. *fix(x)* rounds the elements of *x* to the nearest integers towards zero.

**fclose** - Close file. *st* = *fclose(fid)* closes the file associated with file identifier *fid*, which is an integer value obtained from an earlier call to *fopen()*. *fclose()* returns 0 if successful or -1 if not.

**feof** - Test for end-of-file. *st* = *feof(fid)* returns 1 if the end-of-file indicator for the file with file identifier *fid* has been set, and 0 otherwise. The end-of-file indicator is set when a read operation on the file associated with the *fid* attempts to read past the end of the file.

**fgetl** - read line from file, discard newline character. *tline* = *fgetl(fid)* returns the next line of a file associated with file identifier *fid* as a MATLAB string. The line terminator is NOT included. Use *fget*s() to get the next line with the line terminator INCLUDED. If just an end-of-file is encountered, -1 is returned.

**find** - Returns the linear indexes of non-zero elements in a matrix. *find([0 1 0 1 0])* returns [2 4]. If the first parameter has more than one row, a column vector containing the linear indexes of non-zero elements are returned. An optional second parameter set the maximum number of indexes to return.

**fopen** - Open file. *fid = fopen(filename,permission)* opens the file *filename* in the mode specified by PERMISSION:
'r' - open file for reading
'w' - open file for writing; discard existing contents
'a' - open or create file for writing; append data to end of file
'r+' - open (do not create) file for reading and writing
'w+' - open or create file for reading and writing; discard existing contents
'a+' - open or create file for reading and writing; append data to end of file

**fprintf** - Write formatted data to file. *count = fprintf(fid,format,A,...)* formats the data in the real part of array A (and in any additional array arguments), under control of the specified *format* string, and writes it to the file associated with file identifier *fid*. *count* is the number of bytes successfully written. *fid* is an integer file identifier obtained from *fopen*(). It can also be 1 for standard output (the screen) or 2 for standard error. If *fid* is omitted, output goes to the screen. *format* is a string containing ordinary characters and/or C language conversion specifications. Conversion specifications involve the character %, optional flags, optional width and precision fields, optional subtype
specifier, and conversion characters d, i, o, u, x, X, f, e, E, g, G, c, and s. The special formats \n, \r, \t, \b, \f can be used to produce linefeed, carriage return, tab, backspace, and formfeed characters respectively. Use \\ to produce a backslash character and %% to produce the percent character.

**global** - Define global variable. *global X Y Z* defines *X*, *Y*, and *Z* as global in scope (scope can be functions/programs).

**input** - Read a value from the keyboard and into a variable. *answer=input(str)* prints str as a prompt, reads a number and assigns it to *answer*. If character string is to be read, use the optional second parameter ‘s’.

**isempty** - Determine whether array is empty This MATLAB function returns logical 1 (true) if A is an empty array and logical 0 (false) otherwise. *TF = isempty(A)*

**length** - The length of vector. *length(X)* returns the length of vector X. It is equivalent to *max(size(X))* for non-empty arrays and 0 for empty ones. use size(X,1) and size(X,2) etc to get the size in specific dimensions.

**load** - Loads data from filename. *load(filename)* loads data from filename. If filename is a MAT-file, then *load(filename)* loads variables in the MAT-File into the MATLAB® workspace. If filename is an ASCII file, then *load(filename)* creates a double-precision array containing data from the file.

**max** - finds the highest element in a vector, or the highest element in each column of a matrix.

**min** - finds the lowest element in a vector, or the lowest element in each column of a matrix.

**mod** - Modulus after division. *mod(x,y)* is x - n.*y where *n = floor(x./y)* if y ~= 0.

**num2str** - Convert numbers to a string.

**randi** - Pseudorandom integers from a uniform discrete distribution. *R = randi(IMAX,N)* returns an N-by-N matrix containing pseudorandom integer values drawn from the discrete uniform distribution on *1:IMAX*. Further: *randi(IMAX,M,N)* or *randi(IMAX,[M,N])* returns an M-by-N matrix.

**rem** - Remainder after division. *rem(x,y)* is *x - n.*y* where *n = fix(x./y)* if y ~= 0.

**repmat** - returns a matrix constructed from concatenating its first parameter multiple times. For instance *repmat(foo, 3, 4)* replicates *foo* three times vertically and four times horisontally. The function *repmat('ab',1,3)* returns 'ababab'.

**round** - Rounds to nearest decimal or integer. *Y = round(X)* rounds each element of *X* to the nearest integer. If an element is exactly between two integers, the round function rounds away from zero to the integer with larger magnitude. *Y = round(X,N)* rounds to N digits

**size** - The size of array. *D = size(X)*, for M-by-N matrix X, returns the two-element row vector. D = [M,N] containing the number of rows and columns in the matrix. Further, *size(X,D)* gives the size of *X* in dimention *D*, where 1 means number of rows and 2 means number of columns.

**sortrows** - Sort array rows. This MATLAB function sorts the rows of a matrix, table or cell arry in ascending order, based on values in first column: *B = sortrows(A)*. More generally: *B = sortrows(A [, column [, order]])*. Here, *column* can be a vector of integers, where the second (third etc) element gives the secondary (tertiary etc) column that determine the sort order for elements that have the same value in the primary sort colum. If *order* is present, it must be either 'ascending' or 'descending' to specify the sorting order.

**sprintf** - Basically the same as *fprintf()* without the first, optional parameter fid, but instead of printing to the standard output, it returns the constructed string value.

**sscanf** - Extracts values from a string according to a format string. Opposite of *fprintf()*. *A=sscanf('12/11-2014','%d/%d-%d')* returns a column vector containing the values 12, 11, and 2014.

**strcmp** - Compare strings. *TF = strcmp(S1, S2)* compares the strings *S1* and *S2* and returns logical 1 (true) if they are identical, and returns logical 0 (false) otherwise.

**strsplit** - Splits the first (string) parameter into a cell array of substrings, according to the delimiter string given as the second parameter. *strsplit('one, two, three', ', ')* results in {'one', 'two', 'three'}. Multiple alternative delimiters can be specified using a cell array as the second parameter.

**strtok** - separates the first token of a string from the rest of that string.
*[token, rest] =strtok (' first second', delim)* sets *token* to 'first' and *rest* to ' second'. The optional parameter *delim* contains a list of delimiter characters – where the space character is default. Any delimiter characters before the first token are ignored, but delimitors between first token and the rest of the string is preserved at the front of *rest*.

**str2num** - Convert string matrix to numeric array. *X = str2num(S)* converts a character array representation of a matrix of numbers to a numeric matrix. For example, if *S=['12'; '34']* then *str2num(S)* => [ 12; 34 ]. Further, if *S='abc'* then *str2num(S)=> []*

**sum** - The sum of elements. *S = sum(X)* is the sum of the elements of the vector *X*. If *X* is a matrix, *S* is a row vector with the sum over each column.