NTNU
Kunnskap for en bedre verden

Department for Computer Science

Continuation examination in TDT4105 Information Technology - Introduction

Contact during the exam: Rune Sætre Mobil: 452 18 103
Anders Christensen Mobil: 918 97 181

Exam date: **2017-08-**

Exam time (from-to): **09:00 – 13:00**

Allow aids: **Specified, simple calculator**

Other information:

The exam contains 4 problems. A percentage score is given to show how much each problem and sub-problem counts when the exams are graded. Read through all the problems before you start solving them. Be smart and make good use of your time! If you feel the problems are not fully specified, please write your assumptions explicitly.

Answer briefly and clearly, and write so that the text is easy to read. If the text is ambiguous or longer than necessary, points will be deducted.

Language: **English**

Number of pages: **2017 (including front-page, forms and appendix)**

Content:

- Problem 1: Multiple Choice Questions (25%)
- Problem 2: Programming Price war: (25%)
- Problem 3: Programming Large screen: (30%)
- Problem 4: Understanding code (20%)
- Appendix: Useful functions
- Forms for answering multiple choice questions (2 forms)

Informasjon om trykking av eksamensoppgave
Originalen er:

1-sidig ☐ 2-sidig ☐
sort/hvit ☐ farger ☐

Controlled by:

July 28, 2017

Date Sign

Problem 1: Multiple Choice Questions (25%)

Use the two enclosed forms to solve this exercise (take one home). You can get a new form if you need it. Only one answer is completely correct. For each question, a correct answer counts 1 point. Wrong answer or more than one answer counts -1/2 point. No answer counts 0 points. You get no less than 0 points total for this problem.

1. How many *bytes* do you need to represent a full HD picture (1920x1080) in black & white?
   a. 86 400.
   b. 207 360.
   c. 259 200.
   d. 2 073 600.

2. What is the name of the circuit board in a PC connecting CPU, memory, graphics card and other additional functionality?
   a. PC-card.
   b. Motherboard.
   c. Multi Core Board.
   d. Daughterboard.

3. What does the term Random Access Memory (RAM) mean?
   a. Data is read/written directly independent of where in memory it is located.
   b. Data is read/written sequentially in memory.
   c. It is random what units that have access to various parts of the memory.
   d. The speed of reading/writing of data in memory is random.

4. What is the main difference between Primary and Secondary memory?
   a. Secondary memory is always faster than Primary.
   b. Secondary memory works as a backup if Primary memory stops working.
   c. Secondary memory is permanent, while Primary is temporary.
   d. Primary memory is cheaper per Megabyte than Secondary.

5. Which statement is NOT CORRECT regarding photolithography?
   a. Is used to create integrated circuits (ICs).
   b. The cost and amount of work is the same independent of how complicated the circuits are.
   c. The process enables to put more layers of circuits on top of each other.
   d. RGB is used to expose various layers such as photoresist (blue), unprotected metal (green), and other layers (red).

6. What description fits best for a transistor?
   a. Converts from an analog to a digital signal.
   b. Converts from a digital to an analog signal.
   c. Works like a switch that can be controlled by electrical power signals.
   d. Transfers a digital signal from one physical unit to another.

7. Which alternative describes the ”Fetch/Execute Cycle” best?
   Abbreviations are given in alphabetical order: DF = Data Fetch, EX = Instruction Execution, IF = Instruction Fetch, ID = Instruction Decode, RR = Result Return
   a. IF, ID, DF, EX, RR
   b. DF, IF, ID, EX, RR
   c. IF, ID, EX, DF, RR
   d. RR, DF, IF, ID, EX

8. What is the main task of the ALU?
   a. Fetch and execute instructions.
   b. Connect input and output units.
   c. Execute computing operations.
   d. Control the Program Counter.

9. What is the result of the binary addition of 10101+10101?
   a. 101010.
   b. 101000.
   c. 100000.
   d. 110001.

10. What is the binary number that corresponds to the hexadecimal number D020?
    a. `1101 0010.`
    b. `110 101 010 000.`
    c. `0000 0010 0000 1101.`
    d. `1101 0000 0010 0000.`

11. If ’OSTE’ coded in ASCII is ’`0100 1111 0101 0011 0101 0100 0100 0101`’, what will ’POP’ coded in ASCII be?
    a. ’`0100 1111 0101 0011 0101 0100`’.
    b. ’`0101 0011 0101 0100 0100 0101`’.
    c. ’`0100 1111 0101 0000 0101 0000`’.
    d. ’`0101 0000 0100 1111 0101 0000`’.

12. What is the Nyquist rule about digital recording?
    a. The sampling frequency must be half of the fastest audio frequency.
    b. The sampling frequency must be twice as fast as the fastest audio frequency.
    c. The sampling frequency must be the same as the fastest audio frequency.
    d. The sampling frequency must be 4410Hz.

13. Which statement is NOT CORRECT regarding JPEG?
    a. Picture files in the JPEG-format are less in size than non-compressed picture files.
    b. The JPEG-format uses compression with loss of picture quality.
    c. The JPEG-format is best suited for pictures with simple computer graphics.
    d. There is a direct relationship between picture quality and compression.

14. What does the abbreviation ISP in the textbook stand for?
    a. Internal Storage Protocol.
    b. Internet Service Provider.
    c. Integrated Software Process.
    d. Illustrated Software Plan.

15. I which layer of the TCP/IP reference model does you find HTTP, SMTP, and FTP?
    a. The Application layer.
    b. The Transport layer.
    c. The Network layer.
    d. The Link layer.

16. What is a packet in relation to computer networks?
    a. A block of data with fixed length sent through the network, from a deliverer to a receiver.
    b. A data message with varied length containing all the data sent from a deliverer to a receiver.
    c. A file which is compressed before it is sent over the network to a receiver.
    d. A encrypted file sent over the network, which must be unpacked before it can be used by the receiver.

17. What is a protocol in relation to computer networks?
    a. A contract between the owner of the network and the institution using the network.
    b. A register where all network traffic is stored to be checked by the government.
    c. A set of communication rules for exchange of data.
    d. A compression algorithm which makes it more efficient to send data over network.

18. What is the algorithmic complexity for binary search?
    a. $\Theta(\log n)$.
    b. $\Theta(n)$.
    c. $\Theta(n \log n)$.
    d. $\Theta(n^2)$.

19. What is the algorithmic complexity for ”brute force” (travelling salesman problem)?
    a. $\Theta(n^2)$
    b. $\Theta(n^3)$
    c. $\Theta(2^n)$
    d. $\Theta(n!)$

20. What is a disadvantage of incremental development within software engineering?
    a. Hard to manage changes during the process.
    b. All requirements must be specified in advance.
    c. More difficult for project managers to control deliveries to measure progress.
    d. Only works for big projects.

Problem 2 Programming Price war (25%)

You can assume that all functions receive valid arguments (inputs). You can use functions from sub-problems even you have not solved these problems yourself.

In this problem, you shall create software to compare prices of chosen products from various stores[1]. The starting-point for this comparison is a text file where each line consists of three elements divided by tab (`\t`): Name of store, Name of product, and Price (see text box). Note that this text file might vary in terms of number of stores and number of products to be compared.

pricewar.txt

Rema	Milk	14.50
Rema	Pepsi Max	20.00
Extra	Milk	14.20
Kiwi	Pepsi Max	20.50
Extra	Pepsi Max	19.50
Rema	Banana	12.50
Kiwi	Milk	13.00
Rema	Juice	29.30
Extra	Juice	23.00
Rema	Chocolate	14.00
Extra	Chocolate	13.30
Kiwi	Chocolate	13.00
Kiwi	Banana	10.50
Extra	Banana	11.00

Problem 2a (5%)

Write the function `file_to_list` which has one input parameter `filename`. This function will read a text file with the name `filename` and return a table (), where each row contains name of store, name of product, and price of product. Note that the price of product should be represented as .

Example of calling the function with the file 'pricewar.txt' as shown above:

```
>> dataList = file_to_list('pricewar.txt')
dataList =
    'Rema'         'Milk'          [14.5000]
    'Rema'         'Pepsi Max'     [     20]
    'Extra'        'Milk'          [14.2000]
    'Kiwi'         'Pepsi Max'     [20.5000]
    'Extra'        'Pepsi Max'     [19.5000]
    'Rema'         'Banana'        [12.5000]
    'Kiwi'         'Milk'          [     13]
    'Rema'         'Juice'         [29.3000]
    'Extra'        'Juice'         [     23]
    'Rema'         'Chocolate'     [     14]
    'Extra'        'Chocolate'     [13.3000]
    'Kiwi'         'Chocolate'     [     13]
    'Kiwi'         'Banana'        [10.5000]
    'Extra'        'Banana'        [     11]
    'Kiwi'         'Juice'         [27.5000]
    'Bunnpris'     'Milk'          [     13]
    'Bunnpris'     'Pepsi Max'     [21.5000]
    'Bunnpris'     'Banana'        [15.9000]
    'Bunnpris'     'Juice'         [     26]
    'Bunnpris'     'Chocolate'     [12.5000]
>>
```

[1] The given prices are not real, and do not represent real prices from the named stores.

Problem 2b (4%)
Write the function `list_stores` which has `dataList` as input parameter. `dataList` is a table () the one returned from the function `file_to_list` in Problem 2a. The function shall return a complete list of stores it finds in the table `dataList`. Each store should just one entry in the list. Note that you never know what stores the list will contain. The order of stores should be the same as the order they in the table `dataList`.

Example of calling the function with the file 'pricewar.txt' as shown above:

```
>> dataList = file_to_list('pricewar.txt');
>> storeList = list_stores(dataList)
storeList =
    'Rema'    'Extra'    'Kiwi'    'Bunnpris'
>>
```

Problem 2c (5%)
Write the function `sum_prices_stores` which has the input parameters `dataList` and `storeList` (from Problem 2a and 2b). The function shall return a list of the total sum for all products per store. The order of total sums should be the same as the order of stores in `storeList`.

Example of calling the function with the file 'pricewar.txt' as shown above. The result is the sum of prices for the stores Rema, Extra, Kiwi and Bunnpris (in same order as in Problem 2b).

```
>> dataList = file_to_list('pricewar.txt');
>> storeList = list_stores(dataList);
>> sumStores = sum_prices_stores(dataList,storeList)
sumStores =
   90.3000   81.0000   84.5000   88.9000
>>
```

Problem 2d (6%)
Write the function `rank_stores` which has the input parameters `storeList` and `sumStores` (from Problem 2b and 2c). The function shall return a list with the names of the stores sorted from the store with lowest prices to the store with highest prices.

Example of calling the function with the file 'pricewar.txt' as shown above. Note that before `rank_stores` is executed, the list of stores is in the same order as in the text file 'pricewar.txt'. After executing the function `rank_stores`, the order is sorted by the stores with lowest prices.

```
>> dataList = file_to_list('pricewar.txt');
>> storeList = list_stores(dataList)
storeList =
    'Rema'    'Extra'    'Kiwi'    'Bunnpris'
>> sumStores = sum_prices_stores(dataList,storeList);
>> storeList = rank_stores(storeList,sumStores)
storeList =
    'Extra'    'Kiwi'    'Bunnpris'    'Rema'
>>
```

Problem 2e (5%)
Write the function `store_analysis` which has one input parameter `filename`. The function shall load a file with the file name `filename`, and then print the sum of the products for each store, and then print the ranking of stores sorted by the products in the file `filename` which are cheapest. The function shall not return anything, but give a print out to the screen as shown below.

Example of calling the function with the file 'pricewar.txt' as shown above.

```
>> store_analysis('pricewar.txt')
The total price for shopping per store is:
Rema : 90.3 kr
Extra : 81.0 kr
Kiwi : 84.5 kr
Bunnpris : 88.9 kr

The ranking of stores according to prices is:
1 Extra
2 Kiwi
3 Bunnpris
4 Rema
>>
```

Problem 3 Programming Big Screen (30%)

In this problem, you should help *Katpiss Everbeen* to create functions to show text on a big screen for large events. The big screen can show 6 lines with text, where each line consists of 30 characters as shown in `Figure 1`.

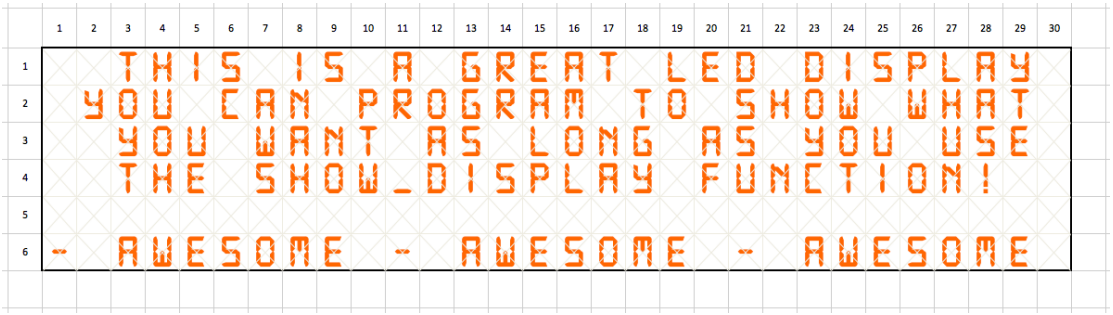


Figure 1 Big Screen

The big screen comes with the function `show_display` to present text on the screen that you can use in your code. The function has one input parameter `content`, which is a six , where each is a text string on 30 characters. If you try to call the function with a list with wrong dimensions, nothing will be shown on the large screen, and the function will return the error message `"Error: Wrong dimensions"`. The big screen can only shown upper case letters, the function `show_display` will make sure to translate to upper case letters if needed.

In this problem, it is recommended to reuse functions from other sub-problems where it is natural. You can use functions from other sub-problems even if you were not able to sole the problem.

Problem 3a (4%)

Write the function `enter_line` which has two input parameters `prompt` and . The function should ask the user to input a sentence which will be returned as a text string. The sentence should be of length specified by the input parameter . If the sentence is not a specified length, the function should give the error message: ”The text must be `[]` characters long”, and continue to ask for a new sentence until the user has given one with correct length. The parameter `prompt` specifies what the user will be asked about.

Example of calling the function (user input is written in **bold font**):

```
>> enter_line('Enter line 1: ',30)
Enter line 1: ITGK is the best!
The text must be 30 characters long
Enter line 1: This is a test on writing nicely and cooly!
The text must be 30 characters long
Enter line 1: This is a test on writing nice
ans =
This is a test on writing nice
>>
```

Problem 3b (4%)
Write the function `adjust_string` which has two input `text` and . The function shall return a new version of the text string `text` with length . If the string `text` has more characters than , the remaining text should be cut. If the string `text` has less characters than , the text should be center adjusted and space must be added on both sides of the text to make the length of the string .

Example of calling the function `adjust_string` as shown below:

```
>> adjust_string('This is a test on writing nicely and cooly!',30)
ans =
This is a test on writing nice
>> adjust_string('ITGK is the best!',30)
ans =
      ITGK is the best!
>> adjust_string('ITGK',30)
ans =
             ITGK
>>
```

Problem 3c (3%)
Write a smarter version of the function `enter_line_smart` (from Problem 3a), which has the two input parameters `prompt` and . The function will receive input from the user, using the text of asking specified with `prompt`, and return a text string of length . If the text the user writes is longer than the remaining text shall be cut, and if the text the user writes is shorter, then text should be center adjusted and filled with spaces to make a text string of number of characters.

Example of calling the function `enter_line_smart` as shown below:

```
>> enter_line_smart('Enter line 1: ',30)
Enter line 1: ITGK is the best!
ans =
       ITGK is the best!
>> enter_line_smart('Enter line 2: ',30)
Enter line 2: This is a test on writing nicely and cooly!
ans =
This is a test on writing nice
>> enter_line_smart('Enter line 3: ',30)
Enter line 3: ITGK
ans =
             ITGK
>>
```

Problem 3d (4%)
Write the function `enter_show_text` that asks the user about entering six lines with text of 30 characters, which then will be shown on the big screen. The function has no input paramters and does not return anything. If the text the user enters is over 30 characters, the remaining text must be cut. If the text the user enters is less than 30 characters, the text shall be center adjusted and filled with space to make it 30 characters long.

Example of calling the function (user-input is written in **bold font**):

```
>> enter_show_text();
Line 1: Welcome to this great concert!
Line 2: ==============================
Line 3: Check out all the great stuff
Line 4: you can buy at our gift store
Line 5: ITGK plays the best music!
Line 6: ITGK for ever!
>>
```

Følgende blir da vist på storskjermen:

Problem 3e (5%)

Write the function `scroll_display` with the two input parameters `content` and `line`. The function will not return anything. The parameter `content` is a list containing 6 text strings of 30 characters, and the parameter `line` which is an integer between 1 and 6. The function shall display the content from the list `content` on the big screen, where the text on line `line` shall be rotated towards left (scrolled) until the text on this line is back where it started (as shown on the figure down below). The update of the big screen should happen every tenth of a second (0,1 sec). The tet on line `line` will be moved 30 times to the left before the function ends. You can assume that the function will be called with correct arguments (`content` contains 6 string of 30 characters and `line` is an integer between 1 and 6).You can make the time delay by using the function `p(s)` from the library `time`, where `s` specifies number of second delay. Example on usage: `p(0.5)` produces a delay of ½ a second.

Example of calling the function where line 1 rotates (scrolls) towards the left:

```
>> content=['Welcome to this great concert!',
           '==============================',
           'Check out all the great stuff ',
           'you can buy at our gift store ',
           '  ITGK plays the best music!  ',
           '        ITGK for ever!        '];
>> scroll_display(content,1)
```

The following will be displayed on the big screen (extracts from the full execution):

0,1 seconds later:

0,1 seconds later:

… and the end (after 3 seconds):

Problem 3f (10%)
Write the function `display_from_file` with one input parameter `filename`. The function shall read from the text file `filename`, and show the content of the text file on the big screen – six lines at a time. The function will not return anything. If the text of a line in the file is over 30 character, the remaining text must be cut. If the text of a line is less than 30 characters, the text should be center adjusted and filled up with space. The function should have a 10 seconds pause between every time new content is shown on the big screen. You can assume the file has a number of lines which is divided by six.

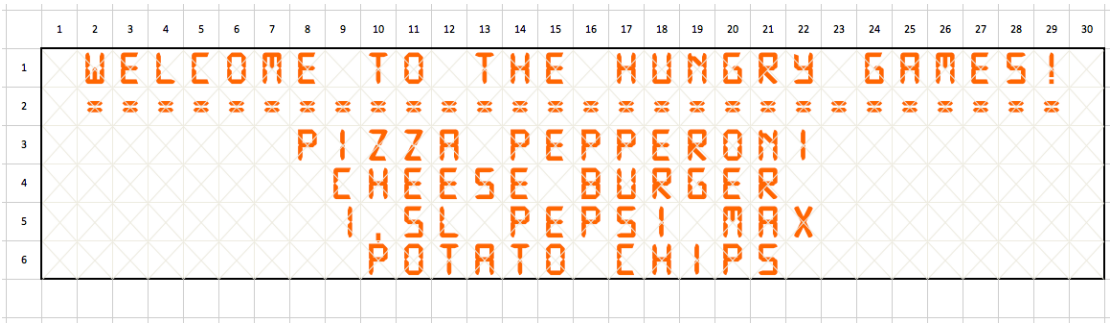
Assume that the content of the text file `message.txt` is:

```
Welcome to the Hungry Games!
============================
Pizza pepperoni
Cheese burger
1,5L Pepsi Max
Potato chips
You will not get any of
these snacks when competing!
Stay hungry and look at all
the stuff you cannot eat!
Happy Hungry Games, and may
the odds be ever in your favor
```

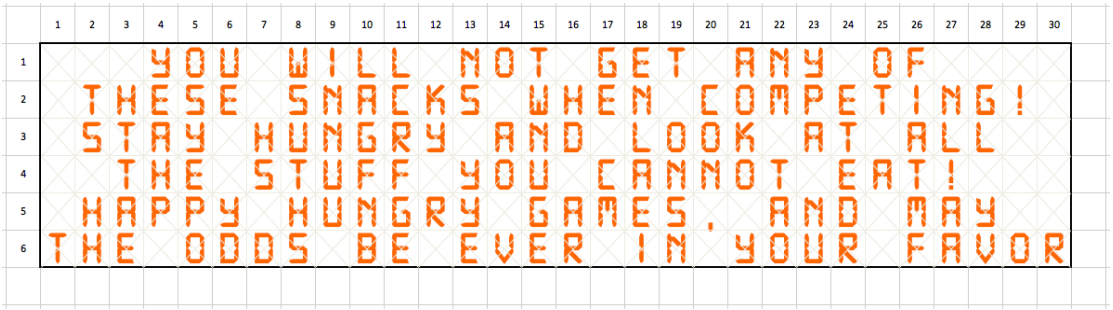
Example of calling the function with the content of `message.txt` as shown above will be:

```
>> display_from_file('message.txt')
```

The following will be displayed on the big screen:



Ten seconds later, the following will be displayed on the big screen:



Problem 4 Understanding of Code (20%)

Problem 4a (5%)
What will be returned when calling the function `myst1('G dg','omd!','dia!')` with the code as shown below (3 %)
Explain with one sentence what the function `myst1()` does? (2 %)

```
function s = myst1(s1,s2,s3)
  s = '';
  for i = 1 : length(s1)
    s = [s s1(i) s2(i) s3(i)];
  end %for
end %function
```

Problem 4b (5%)
Which value will m get when the code below is executed? (3 %)
Explain with one what the function `myst2( )` does? (2 %)

```
function m = main()
    m=[[1,2,3,4,5]
       [2,3,4,5,6]
       [3,4,5,6,7]
       [4,5,6,7,8]
       [5,6,7,8,9]];
    m = myst2(m);
end

function m = myst2(m)
  [r,c] = size(m);
  for y = 1:r
    for x = 1:c
      if y == 1 || y == length(m)
        m(y,x) = 0;
      elseif x == 1 || x == length(m)
        m(y,x) = 0;
      end % if
    end % for x
  end % for y
end % function
```

Problem 4c (5%)
What is returned when executing the function `myst3('xsidrwteasMc hedhfT')` with the code as shown below? (3 %)
Explain with one sentence what `myst3()` does? (2 %)

```
function a = myst3(s)
  a = '';
  for x = length(s):-2:1
    a(end+1) = s(x);
  end % for
end % function
```

Problem 4d (5%)
What is returned when executing the function `myst4(2,1,4)` with the code shown as below? (3%)
Explain with one sentence what the function `myst4()` does? (2%)

```
function x = myst4(x,y,z)
  if y < z
    x = myst4(x*x,y+1,z);
  end %if
end %function
```

Appendix: Possibly useful functions

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

fix
Round towards zero. FIX(X) rounds the elements of X to the nearest integers towards zero.

floor
Round towards minus infinity. FLOOR(X) rounds the elements of X to the nearest integers towards minus infinity.

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 FGETS 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.

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.

pause

pause(n) pauses for n seconds before continuing, where n can also be a fraction. The resolution of the clock is platform specific. Fractional pauses of 0.01 seconds should be supported on most platforms.

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.
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.

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.

sortrows

Sort array rows. This MATLAB function sorts the rows of A in ascending order, based on column. B = sortrows(A). B = sortrows(A, column)

sqrt

Square root. SQRT(X) is the square root of the elements of X.

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.

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, S=['12'; '34'] str2num(S) => [12; 34].
S='abc' 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.

Answer Form for Multiple Choice Questions

Candidate number: _____ Program: _____

Course code: _____ Date: _____

Number of pages: _____ Page: _____

Question	*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				

This page is empty on purpose!

Answer Form for Multiple Choice Questions

Candidate number: _____ Program: _____

Course code: _____ Date: _____

Number of pages: _____ Page: _____

Problem#	***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				