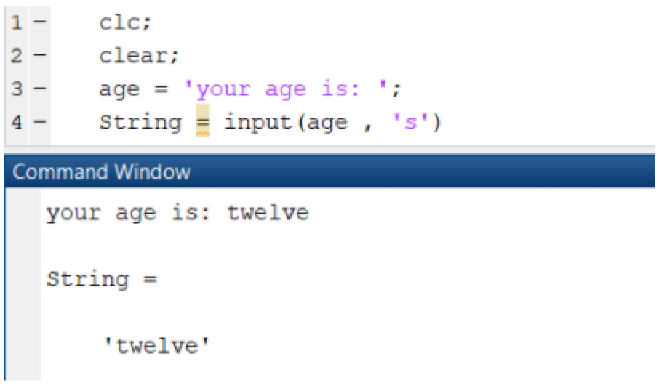
MATLAB – HW1 Upload Date: 31 Oct. 2020

Dorreen Rostami – 97243034
Zahra Hashemi – 97243072

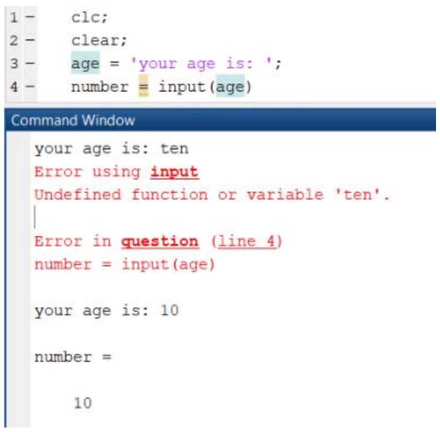
	Syntax	Description	Input	Output
input	x = input(prompt)	x = input(prompt) displays the text in prompt and waits for the user to input a value and press the **Return** key. The user can enter expressions, like pi/4 or rand(3), and can use variables in the workspace. • If the user presses the **Return** key without entering anything, then input returns an empty matrix. • If the user enters an invalid expression at the prompt, then MATLAB® displays the relevant error message, and then redisplays the prompt.	prompt – Text displayed to the user, specified as a character vector.	x – Result calculated from input, returned as an array
	str = input(prompt,'s')	str = input(prompt,'s') returns the entered text, without evaluating the input as an expression.		Str – Exact text of the input, returned as a character vector.



```
clc;
clear;
age = 'your age is: ';
String = input(age , 's')
```

Command Window

```
your age is: twelve

String =

    'twelve'
```

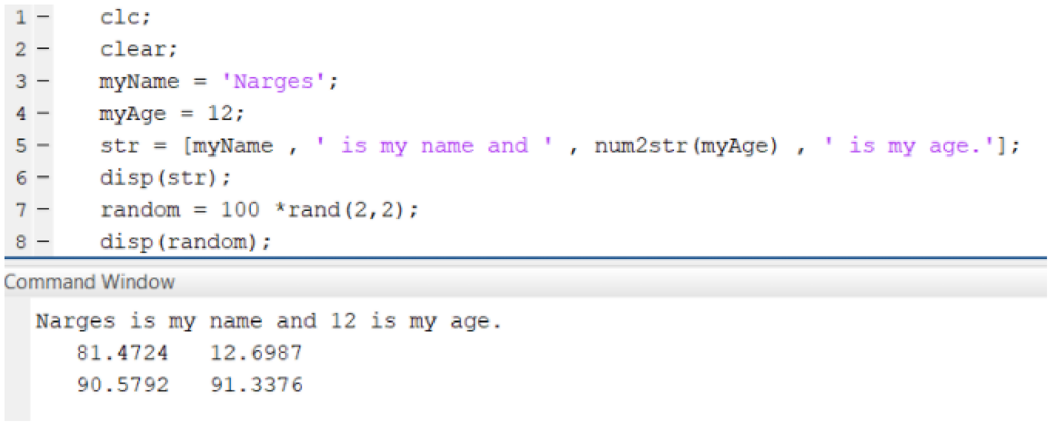


```
clc;
clear;
age = 'your age is: ';
number = input(age)
```

Command Window

```
your age is: ten
Error using input
Undefined function or variable 'ten'.

Error in question (line 4)
number = input(age)

your age is: 10

number =

    10
```

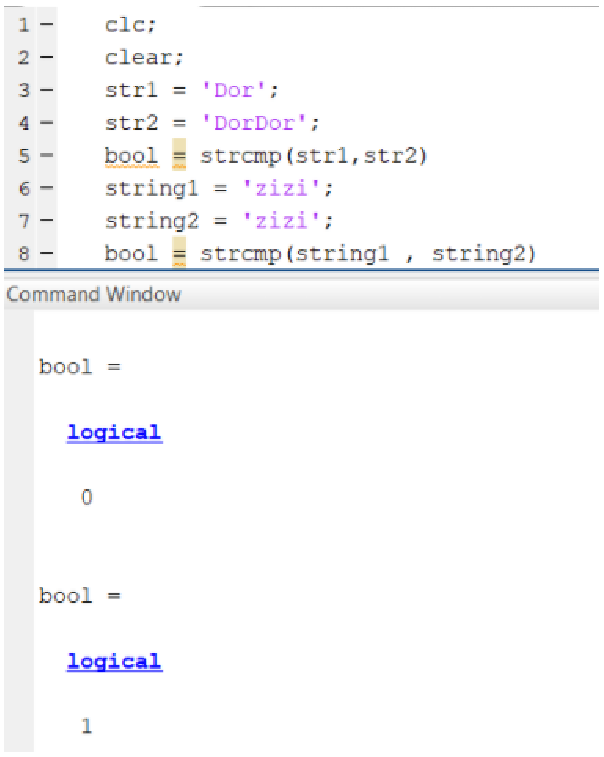
	Syntax	Description	Input
display	disp(x)	disp(X) displays the value of variable X without printing the variable name. Another way to display a variable is to type its name, which displays a leading “X =” before the value. If a variable contains an empty array, disp returns without displaying anything.	X – input array



```
clc;
clear;
myName = 'Narges';
myAge = 12;
str = [myName , ' is my name and ' , num2str(myAge) , ' is my age.'];
disp(str);
random = 100 *rand(2,2);
disp(random);
```

```
Narges is my name and 12 is my age.
   81.4724   12.6987
   90.5792   91.3376
```

	Syntax	Description	Input	Output
strcmp	tf = strcmp(s1,s2)	tf = strcmp(s1,s2) compares s1 and s2 and returns 1 (true) if the two are identical and 0 (false) otherwise. Text is considered identical if the size and content of each are the same. The return result tf is of data type logical. The input arguments can be any combination of string arrays, character vectors, and cell arrays of character vectors.	s1, s2 – Input text, with each input specified as a character vector, a character array, a cell array of character vectors, or a string array. The order of the inputs does not affect the comparison results.	tf – True or false result, returned as a 1 or 0 of data type logical.

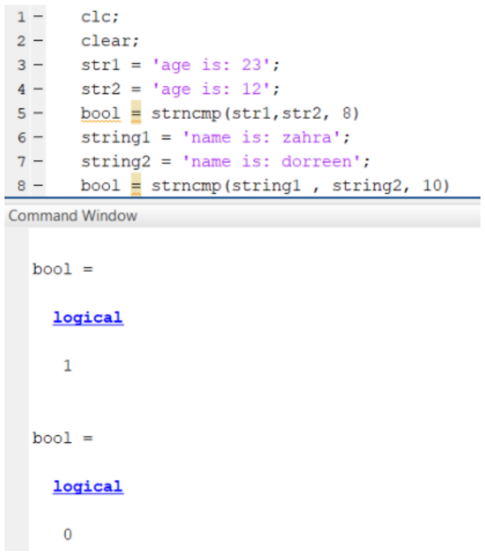


```
clc;
clear;
str1 = 'Dor';
str2 = 'DorDor';
bool = strcmp(str1,str2)
string1 = 'zizi';
string2 = 'zizi';
bool = strcmp(string1 , string2)
```

Command Window

```
bool =

  logical

   0


bool =

  logical

   1
```

	Syntax	Description	Input	Output
strncmp	tf = strncmp(s1,s2,n)	tf = strncmp(s1,s2,n) compares the first n characters of s1 and s2. The function returns 1 (true) if the two are identical and 0 (false) otherwise. Text is considered identical if the size and content of each are the same, up to the first n characters of each piece of text. The return result tf is of data type logical. The first two input arguments can be any combination of string arrays, character vectors, and cell arrays of character vectors.	s1, s2 – Input text, with each input specified as a character vector, a character array, a cell array of character vectors, or a string array. The order of the inputs does not affect the comparison results. n – Number of characters to compare, specified as an integer.	tf – True or false result, returned as a 1 or 0 of data type logical.

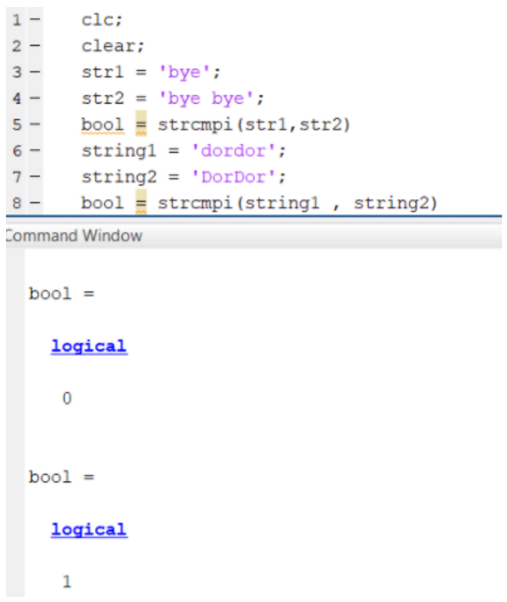


```
clc;
clear;
str1 = 'age is: 23';
str2 = 'age is: 12';
bool = strncmp(str1,str2, 8)
string1 = 'name is: zahra';
string2 = 'name is: dorreen';
bool = strncmp(string1 , string2, 10)
```

Command Window

```
bool =

  logical

   1


bool =

  logical

   0
```

	Syntax	Description	Input	Output
strcmpi	tf = strcmpi(s1,s2)	tf = strcmpi(s1,s2) compares s1 and s2, ignoring any differences in letter case. The function returns 1 (true) if the two are identical and 0 (false) otherwise. Text is considered identical if the size and content of each are the same, aside from case. The return result tf is of data type logical. The input arguments can be any combination of string arrays, character vectors, and cell arrays of character vectors.	s1, s2 – Input text, with each input specified as a character vector, a character array, a cell array of character vectors, or a string array. The order of the inputs does not affect the comparison results.	tf – True or false result, returned as a 1 or 0 of data type logical.

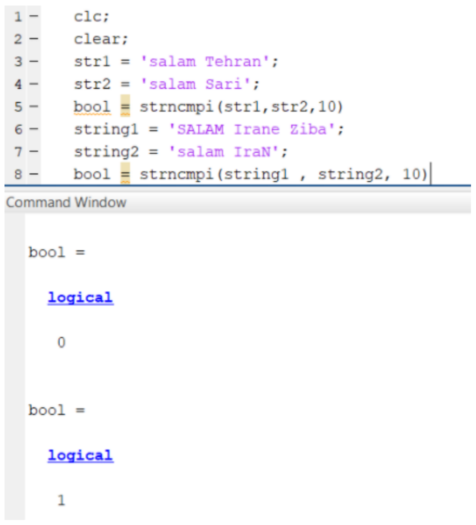


```
clc;
clear;
str1 = 'bye';
str2 = 'bye bye';
bool = strcmpi(str1,str2)
string1 = 'dordor';
string2 = 'DorDor';
bool = strcmpi(string1 , string2)
```

Command Window

```
bool =

  logical

   0


bool =

  logical

   1
```

	Syntax	Description	Input	Output
strncmpi	tf = strncmpi(s1,s2,n)	tf = strncmpi(s1,s2,n) compares the first n characters of s1 and s2, ignoring any differences in letter case. The function returns 1 (true) if the two are identical and 0 (false) otherwise. Text is considered identical if the size and content of each are the same, up to the first n characters of each piece of text, ignoring case. The return result tf is of data type logical. The first two input arguments can be any combination of string arrays, character vectors, and cell arrays of character vectors.	s1, s2 – Input text, with each input specified as a character vector, a character array, a cell array of character vectors, or a string array. The order of the inputs does not affect the comparison results. n – Number of characters to compare, specified as an integer.	tf – True or false result, returned as a 1 or 0 of data type logical.

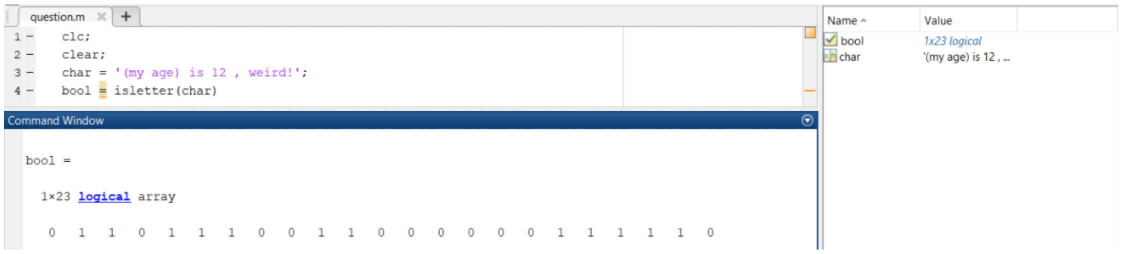


```
clc;
clear;
str1 = 'salam Tehran';
str2 = 'salam Sari';
bool = strncmpi(str1,str2,10)
string1 = 'SALAM Irane Ziba';
string2 = 'salam IraN';
bool = strncmpi(string1 , string2, 10)
```

Command Window

```
bool =

  logical

   0


bool =

  logical

   1
```

	Syntax	Description	Input
isletter	TF = isletter(A)	TF = isletter(A) returns a logical array TF. If A is a character array or string scalar, then the elements of TF are logical 1 (true) where the corresponding characters in A are letters, and logical 0 (false) elsewhere. If A is not a character array or string scalar, then isletter returns logical 0 (false).	A – Input array, specified as a scalar, vector, matrix, or multidimensional array. A can be any data type.



question.m

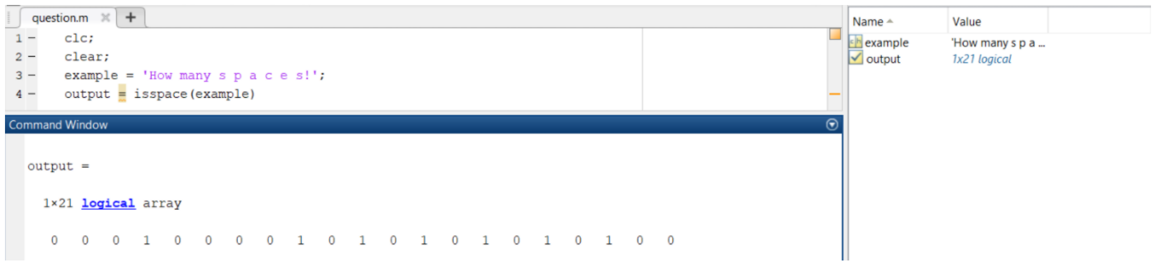
```
clc;
clear;
char = '(my age) is 12 , weird!';
bool = isletter(char)
```

Command Window

```
bool =

  1×23 logical array

   0   1   1   0   1   1   1   0   0   1   1   0   0   0   0   0   0   1   1   1   1   1   0
```

Name	Value
bool	1x23 logical
char	'(my age) is 12 , ...

	Syntax	Description	Input
isspace	TF = isspace(A)	TF = isspace(A) returns a logical array TF. If A is a character array or string scalar, then the elements of TF are logical 1 (true) where corresponding characters in A are space characters, and logical 0 (false) elsewhere. isspace recognizes all Unicode whitespace characters. If A is not a character array or string scalar, then isspace returns logical 0 (false).	A – Input array, specified as a scalar, vector, matrix, or multidimensional array. A can be any data type.



question.m

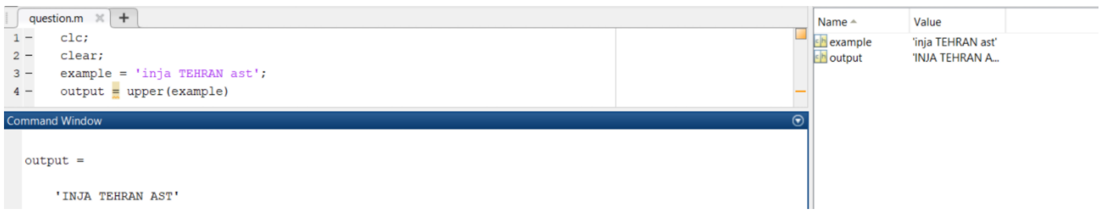
```
clc;
clear;
example = 'How many s p a c e s!';
output = isspace(example)
```

Command Window

```
output =

  1×21 logical array

   0   0   0   1   0   0   0   0   1   0   1   0   1   0   1   0   1   0   1   0   0
```

Name	Value
example	'How many s p a ...
output	1x21 logical

	Syntax	Description	Input
upper	newTxt = upper(txt)	newTxt = upper(txt) converts all lowercase characters in txt to the corresponding uppercase characters and leaves all other characters unchanged.	txt – Input array, specified as a string array, character array, or cell array of character vectors.



question.m

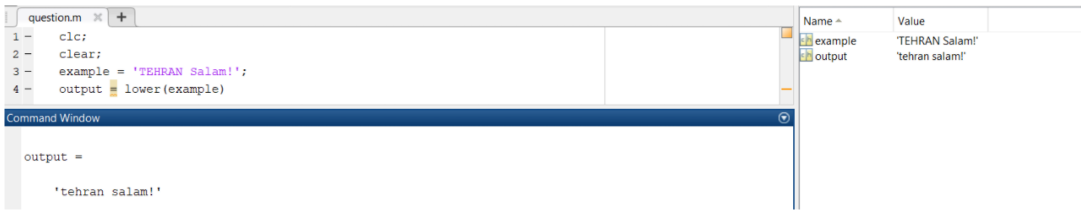
```
clc;
clear;
example = 'inja TEHRAN ast';
output = upper(example)
```

Command Window

```
output =

    'INJA TEHRAN AST'
```

Name	Value
example	'inja TEHRAN ast'
output	'INJA TEHRAN A...

	Syntax	Description	Input
lower	newTxt = lower(txt)	newTxt = lower(txt) converts all uppercase characters in txt to the corresponding lowercase characters and leaves all other characters unchanged.	txt – Input array, specified as a string array, character array, or cell array of character vectors.



question.m

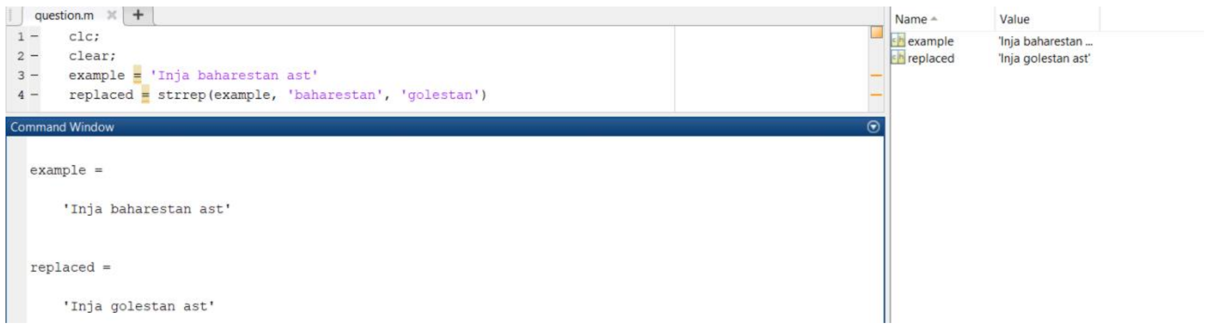
```
clc;
clear;
example = 'TEHRAN Salam!';
output = lower(example)
```

Command Window

```
output =

    'tehran salam!'
```

Name	Value
example	'TEHRAN Salam!'
output	'tehran salam!'

	Syntax	Description	Input
strrep	newStr = strrep(str,old,new)	newStr = strrep(str,old,new) replaces all occurrences of old in str with new.	str – Input text, specified as a string array, character vector, or cell array of character vectors. old –Substring to replace, specified as a string array, character vector, or cell array of character vectors. New – New substring, specified as a string array, character vector, or cell array of character vectors.



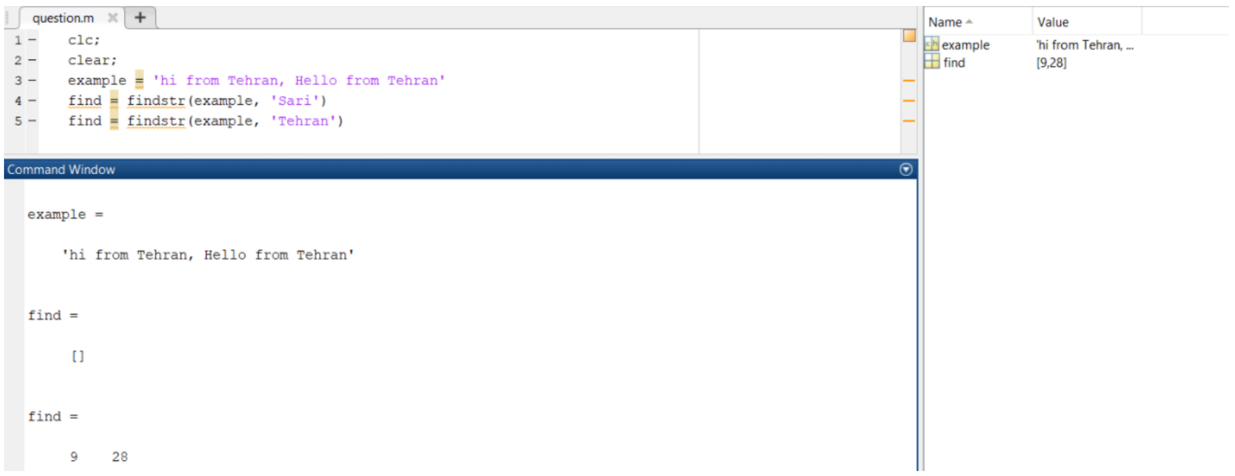
```
clc;
clear;
example = 'Inja baharestan ast'
replaced = strrep(example, 'baharestan', 'golestan')
```

```
example =

    'Inja baharestan ast'


replaced =

    'Inja golestan ast'
```

Name	Value
example	'Inja baharestan ...
replaced	'Inja golestan ast'

	Syntax	Description
findstr	k = findstr(str1, str2)	k = findstr(str1, str2) searches the longer of the two input arguments for any occurrences of the shorter argument, returning the starting index of each such occurrence in the double array k. If no occurrences are found, then findstr returns the empty array, []. The input arguments str1 and str2 can be character vectors or string scalars. The search performed by findstr is case sensitive. Any leading and trailing blanks in either input argument is explicitly included in the comparison. Unlike the contains or strfind functions, the order of the input arguments to findstr is not important. This can be useful if you are not certain which of the two input arguments is the longer one.

- **findstr is not recommended. Use contains or strfind instead.**

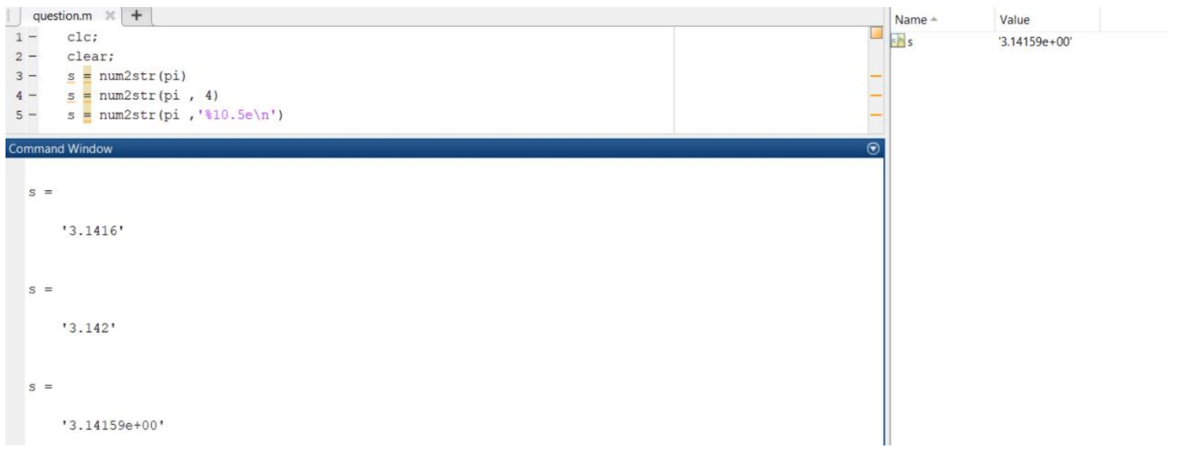


```
clc;
clear;
example = 'hi from Tehran, Hello from Tehran'
find = findstr(example, 'Sari')
find = findstr(example, 'Tehran')
```

```
example =

    'hi from Tehran, Hello from Tehran'


find =

     []


find =

     9    28
```

Name	Value
example	'hi from Tehran, ...
find	[9,28]

	Syntax	Description	Input	Output
num2str	s = num2str(A)	s = num2str(A) converts a numeric array into a character array that represents the numbers. The output format depends on the magnitudes of the original values. num2str is useful for labeling and titling plots with numeric values.	formatSpec – Format of the output fields, specified using formatting operators. formatSpec also can include ordinary text and special characters.	s – Text representation of the input array, returned as a character array.
	s = num2str(A,precision)	s = num2str(A,precision) returns a character array that represents the numbers with the maximum number of significant digits specified by precision.		
	s = num2str(A,formatSpec)	s = num2str(A,formatSpec) applies a format specified by formatSpec to all elements of A.		

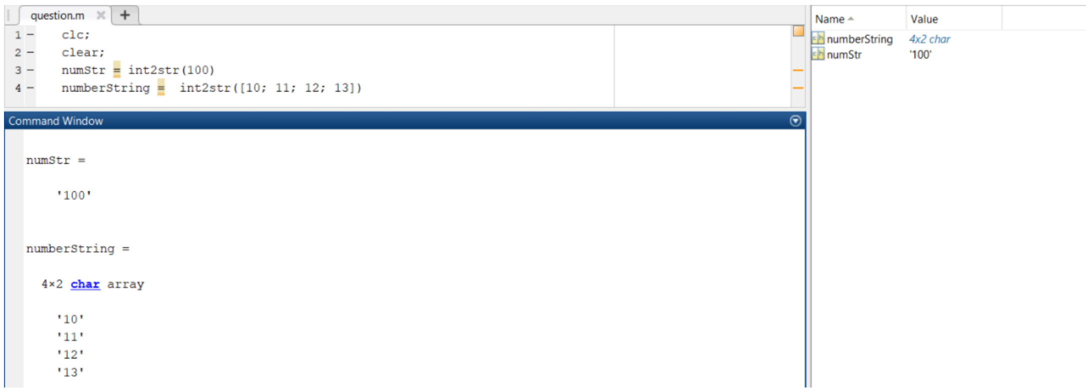


```
clc;
clear;
s = num2str(pi)
s = num2str(pi , 4)
s = num2str(pi ,'%10.5e\n')
```

```
s =

    '3.1416'


s =

    '3.142'


s =

    '3.14159e+00'
```

Name	Value
s	'3.14159e+00'

	Syntax	Description	Input
int2str	chr = int2str(N)	chr = int2str(N) treats N as a matrix of integers and converts it to a character array that represents the integers. If N contains floating-point values, int2str rounds them before conversion.	n – Input array, specified as a numeric matrix.

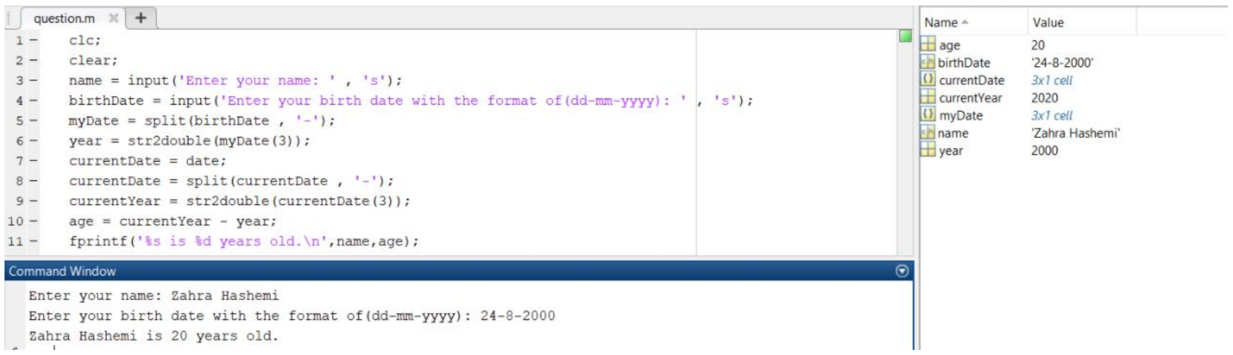


```
clc;
clear;
numStr = int2str(100)
numberString =  int2str([10; 11; 12; 13])
```

```
numStr =

    '100'


numberString =

  4×2 char array

    '10'
    '11'
    '12'
    '13'
```

Name	Value
numberString	4x2 char
numStr	'100'

	Syntax	Description
date	str = date	str = date returns a character vector containing the date in the format, day-month-year, for example, 01-Jan-2014.
now	t = now	t = now returns the current date and time as a serial date number. A serial date number represents the whole and fractional number of days from a fixed, preset date (January 0, 0000).
clock	c = clock	c = clock returns a six-element date vector containing the current date and time in decimal form: [year month day hour minute seconds] The clock function calculates the current date and time from the system time.
	[c tf] = clock	[c tf] = clock returns a second output argument that is 1 (true) if the current date and time occur during Daylight Saving Time (DST) in your system's time zone, and 0 (false) otherwise.



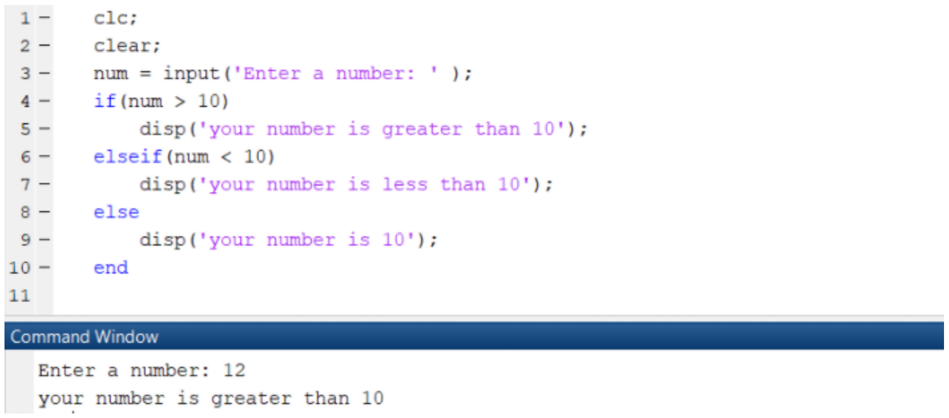
```
clc;
clear;
name = input('Enter your name: ' , 's');
birthDate = input('Enter your birth date with the format of(dd-mm-yyyy): ' , 's');
myDate = split(birthDate , '-');
year = str2double(myDate(3));
currentDate = date;
currentDate = split(currentDate , '-');
currentYear = str2double(currentDate(3));
age = currentYear - year;
fprintf('%s is %d years old.\n',name,age);
```

```
Enter your name: Zahra Hashemi
Enter your birth date with the format of(dd-mm-yyyy): 24-8-2000
Zahra Hashemi is 20 years old.
```

Name	Value
age	20
birthDate	'24-8-2000'
currentDate	3x1 cell
currentYear	2020
myDate	3x1 cell
name	'Zahra Hashemi'
year	2000

	Syntax	Description

if, elseif, else	`if expression`
	`    statements`
	`elseif expression`
	`    statements`
	`else`
	`    statements`
	`end`
		if *expression*, *statements*, end evaluates an expression, and executes a group of statements when the expression is true. An expression is true when its result is nonempty and contains only nonzero elements (logical or real numeric). Otherwise, the expression is false.
		The elseif and else blocks are optional. The statements execute only if previous expressions in the if...end block are false. An if block can include multiple elseif blocks.



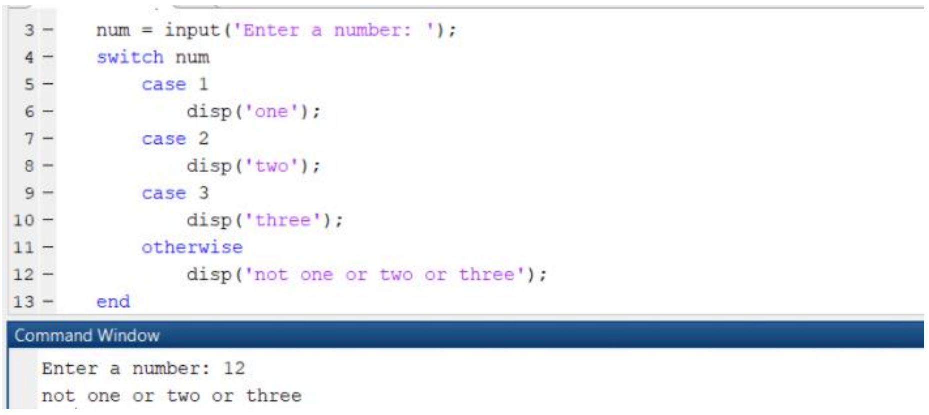
```
clc;
clear;
num = input('Enter a number: ' );
if(num > 10)
    disp('your number is greater than 10');
elseif(num < 10)
    disp('your number is less than 10');
else
    disp('your number is 10');
end
```

Command Window

```
Enter a number: 12
your number is greater than 10
```

	Syntax	Description

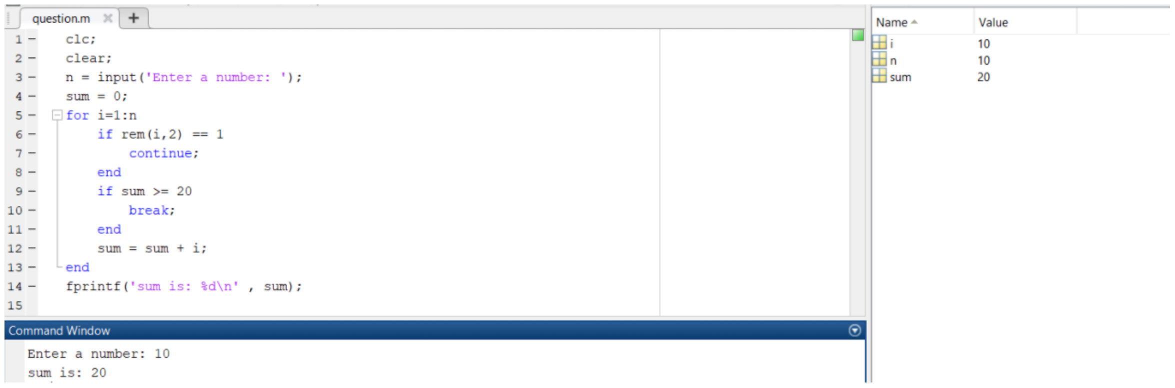
switch, case, otherwise	`switch` *`switch_expression`*
	   `case` *`case_expression`*
	      `statements`
	   `case` *`case_expression`*
	      `statements`
	    `...`
	   `otherwise`
	      `statements`
	`end`
		switch *switch_expression*, case *case_expression*, end evaluates an expression and chooses to execute one of several groups of statements. Each choice is a case.
		The switch block tests each case until one of the case expressions is true. A case is true when:
		• For numbers, *case_expression* == *switch_expression*.
		• For character vectors, strcmp(*case_expression*,*switch_expression*) == 1.
		• For objects that support the eq function, *case_expression* == *switch_expression*.
		• For a cell array *case_expression*, at least one of the elements of the cell array matches *switch_expression*, as defined above for numbers, character vectors, and objects.
		When a case expression is true, MATLAB executes the corresponding statements and exits the switch block.
		An evaluated *switch_expression* must be a scalar or character vector. An evaluated *case_expression* must be a scalar, a character vector, or a cell array of scalars or character vectors.
		The otherwise block is optional. MATLAB executes the statements only when no case is true.



```
num = input('Enter a number: ');
switch num
    case 1
        disp('one');
    case 2
        disp('two');
    case 3
        disp('three');
    otherwise
        disp('not one or two or three');
end
```

```
Enter a number: 12
not one or two or three
```

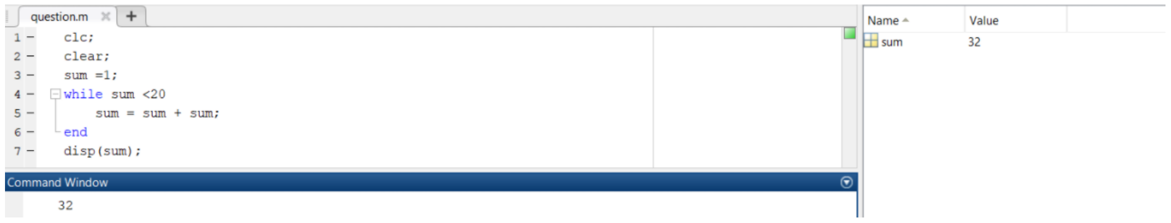
Loops	Syntax	Description
for	`for index = values` `statements` `end`	for *index* = *values*, *statements*, end executes a group of statements in a loop for a specified number of times. *values* has one of the following forms: • *initVal*:*endVal* — Increment the *index* variable from *initVal* to *endVal* by 1, and repeat execution of *statements* until *index* is greater than *endVal*. • *initVal*:*step*:*endVal* — Increment *index* by the value *step* on each iteration, or decrements *index* when *step* is negative. • *valArray* — Create a column vector, *index*, from subsequent columns of array *valArray* on each iteration. For example, on the first iteration, *index* = *valArray*(:,1). The loop executes a maximum of *n* times, where *n* is the number of columns of*valArray*, given by numel(*valArray*(1,:)). The input *valArray* can be of any MATLAB data type, including a character vector, cell array, or struct.
while	`while expression` `statements` `end`	while *expression*, *statements*, end evaluates an expression, and repeats the execution of a group of statements in a loop while the expression is true. An expression is true when its result is nonempty and contains only nonzero elements (logical or real numeric). Otherwise, the expression is false.



```
clc;
clear;
n = input('Enter a number: ');
sum = 0;
for i=1:n
    if rem(i,2) == 1
        continue;
    end
    if sum >= 20
        break;
    end
    sum = sum + i;
end
fprintf('sum is: %d\n' , sum);
```

Name	Value
i	10
n	10
sum	20

```
Enter a number: 10
sum is: 20
```



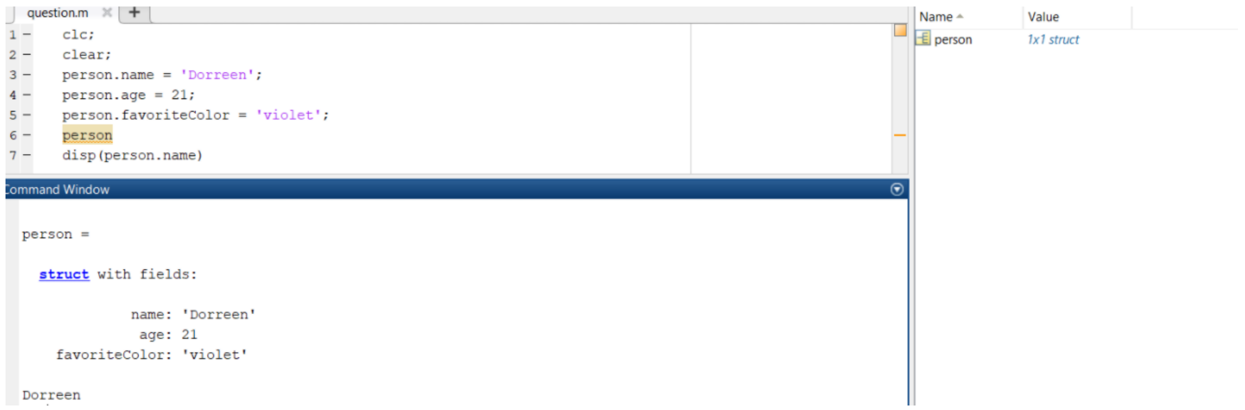
```
clc;
clear;
sum =1;
while sum <20
    sum = sum + sum;
end
disp(sum);
```

Name	Value
sum	32

```
    32
```

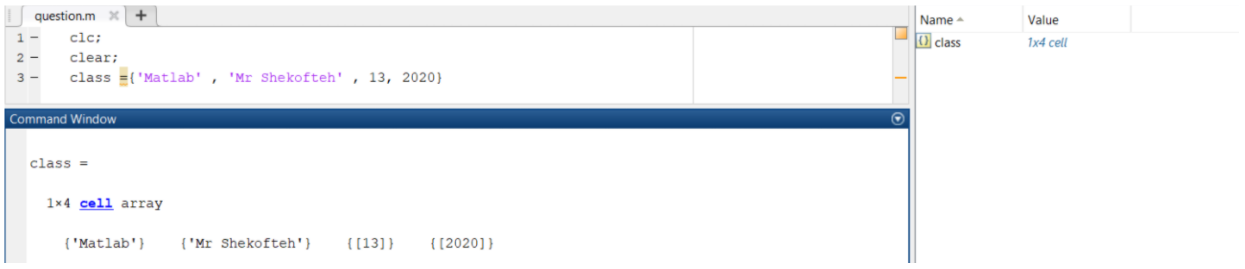
Struct	Description
Structure Array	A *structure array* is a data type that groups related data using data containers called *fields*. Each field can contain any type of data. Access data in a field using dot notation of the form structName.fieldName.

Cell	Description
Cell Array	A *cell array* is a data type with indexed data containers called *cells*, where each cell can contain any type of data. Cell arrays commonly contain either lists of text, combinations of text and numbers, or numeric arrays of different sizes. Refer to sets of cells by enclosing indices in smooth parentheses, (). Access the contents of cells by indexing with curly braces, {}.



```
clc;
clear;
person.name = 'Dorreen';
person.age = 21;
person.favoriteColor = 'violet';
person
disp(person.name)
```

```
person =

  struct with fields:

             name: 'Dorreen'
              age: 21
    favoriteColor: 'violet'

Dorreen
```



```
clc;
clear;
class ={'Matlab' , 'Mr Shekofteh' , 13, 2020}
```

```
class =

  1×4 cell array

    {'Matlab'}    {'Mr Shekofteh'}    {[13]}    {[2020]}
```