Morteza Kazemi | 97243054
Amir Masoud Shaker | 97243081

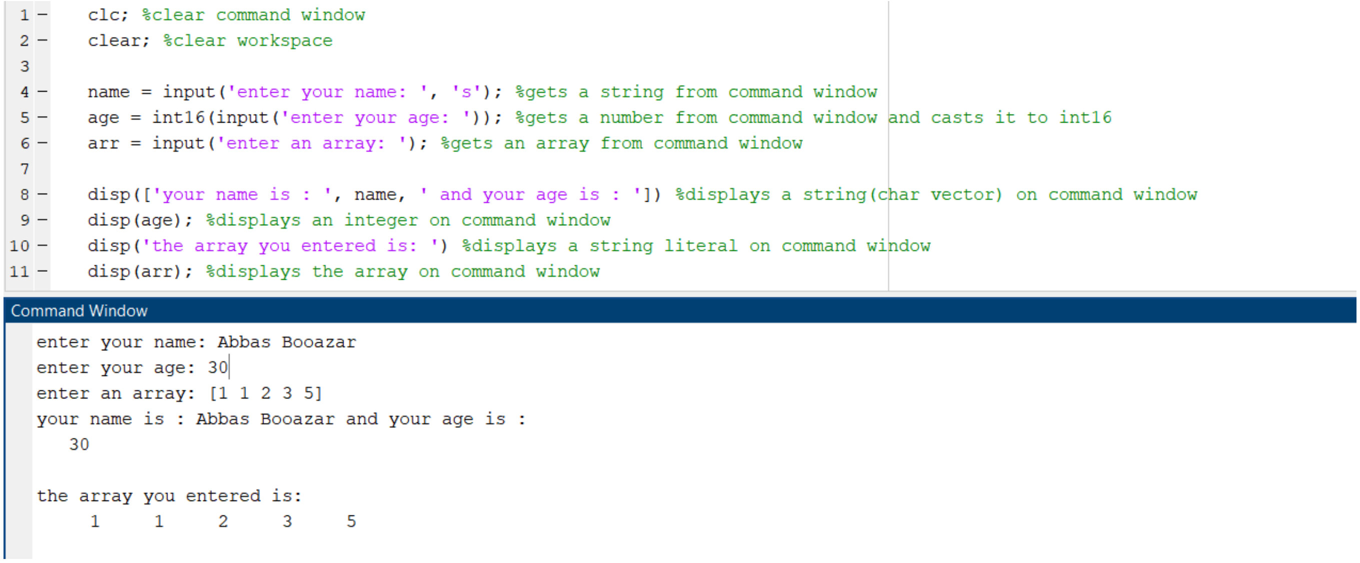
10/31/2021

Homework #1

1-1 : Input and Output in programs:

Name	Syntax	Description
input	x = input(prompt)	This function 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/4or rand(3), and can use variables in the workspace. Note: If the user presses the **Return** key without entering anything, then input returns an empty matrix. Note: If the user enters an invalid expression at the prompt, then MATLAB® displays the relevant error message, and then redisplays the prompt.
	str = input(prompt, 's')	This function returns the entered text, without evaluating the input as an expression.

Name	Syntax	Description
disp	disp(X)	This function 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.



```
clc; %clear command window
clear; %clear workspace

name = input('enter your name: ', 's'); %gets a string from command window
age = int16(input('enter your age: ')); %gets a number from command window and casts it to int16
arr = input('enter an array: '); %gets an array from command window

disp(['your name is : ', name, ' and your age is : ']) %displays a string(char vector) on command window
disp(age); %displays an integer on command window
disp('the array you entered is: ') %displays a string literal on command window
disp(arr); %displays the array on command window
```

```
enter your name: Abbas Booazar
enter your age: 30
enter an array: [1 1 2 3 5]
your name is : Abbas Booazar and your age is :
   30

the array you entered is:
     1     1     2     3     5
```

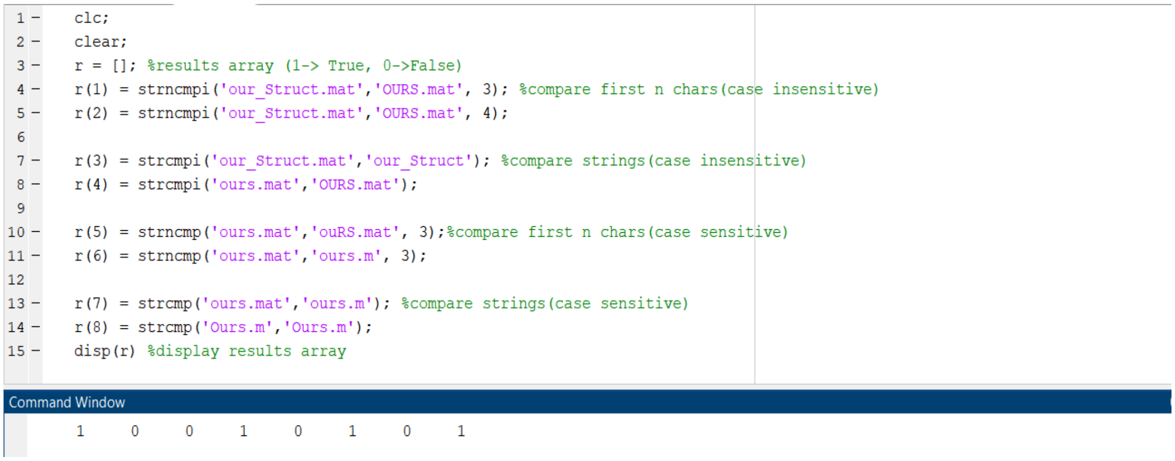
1-2 : String comparison:

Name	Syntax	Description
strcmp	tf = strcmp(s1, s2)	This function 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.

Name	Syntax	Description
strncmp	tf = strncmp(s1, s2, n)	This function compares up to 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 content of each is the same up to the end or the first n characters, whichever comes first. 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.

Name	Syntax	Description
strcmpi	tf = strcmpi(s1, s2)	This function compares up to 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 content of each is the same up to the end or the first n characters, whichever comes first, 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.

Name	Syntax	Description
strncmpi	tf = strncmpi(s1, s2)	This function compares up to 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 content of each is the same up to the end or the first n characters, whichever comes first, 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.



```
clc;
clear;
r = []; %results array (1-> True, 0->False)
r(1) = strncmpi('our_Struct.mat','OURS.mat', 3); %compare first n chars(case insensitive)
r(2) = strncmpi('our_Struct.mat','OURS.mat', 4);

r(3) = strcmpi('our_Struct.mat','our_Struct'); %compare strings(case insensitive)
r(4) = strcmpi('ours.mat','OURS.mat');

r(5) = strncmp('ours.mat','ouRS.mat', 3);%compare first n chars(case sensitive)
r(6) = strncmp('ours.mat','ours.m', 3);

r(7) = strcmp('ours.mat','ours.m'); %compare strings(case sensitive)
r(8) = strcmp('Ours.m','Ours.m');
disp(r) %display results array
```

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

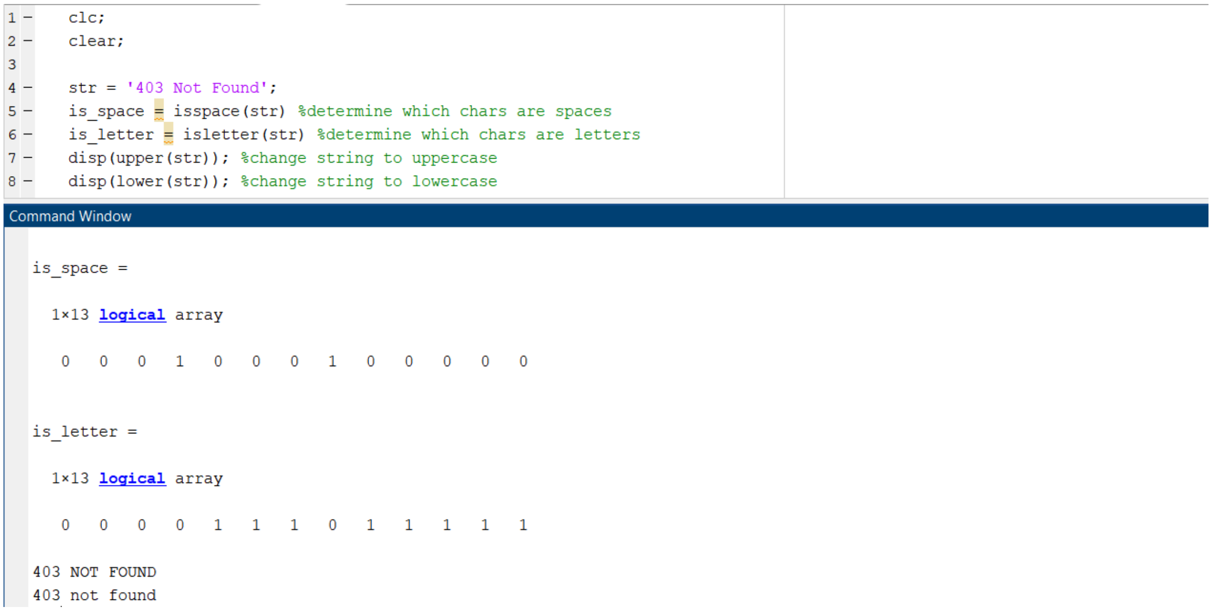
1-3 : Character categorization in strings:

Name	Syntax	Description
isletter	TF = isletter(A)	This function 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).

Name	Syntax	Description
isspace	TF = isspace(A)	This function 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).

Name	Syntax	Description
upper	newStr = upper(str)	This function converts all lowercase characters in str to the corresponding uppercase characters and leaves all other characters unchanged.

Name	Syntax	Description
lower	newStr = lower(str)	This function converts all uppercase characters in str to the corresponding lowercase characters and leaves all other characters unchanged.



```
clc;
clear;

str = '403 Not Found';
is_space = isspace(str) %determine which chars are spaces
is_letter = isletter(str) %determine which chars are letters
disp(upper(str)); %change string to uppercase
disp(lower(str)); %change string to lowercase
```

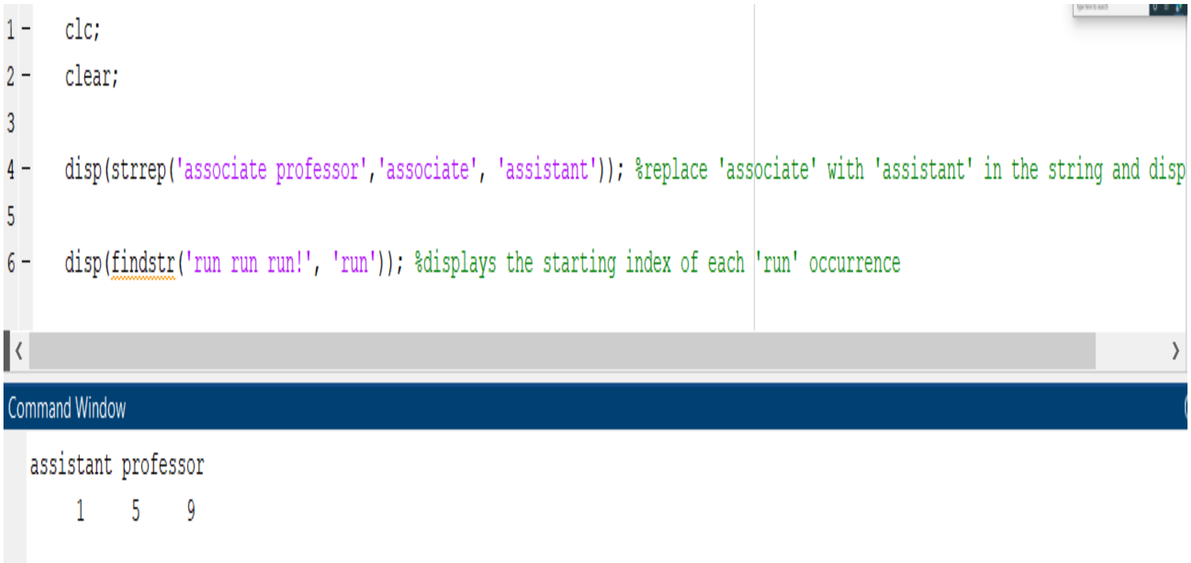
Command Window

```
is_space =

  1×13 logical array

   0   0   0   1   0   0   0   1   0   0   0   0   0


is_letter =

  1×13 logical array

   0   0   0   0   1   1   1   0   1   1   1   1   1

403 NOT FOUND
403 not found
```

1-4 : Searching and replacing strings:

Name	Syntax	Description
strrep	newStr = strrep(str, old, new)	This function replaces all occurrences of old in str with new. If any input argument is a nonscalar string array or cell array of character vectors, then the other input arguments must have compatible sizes.

Name	Syntax	Description
findstr	K = findstr(str1, str2)	This function 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.



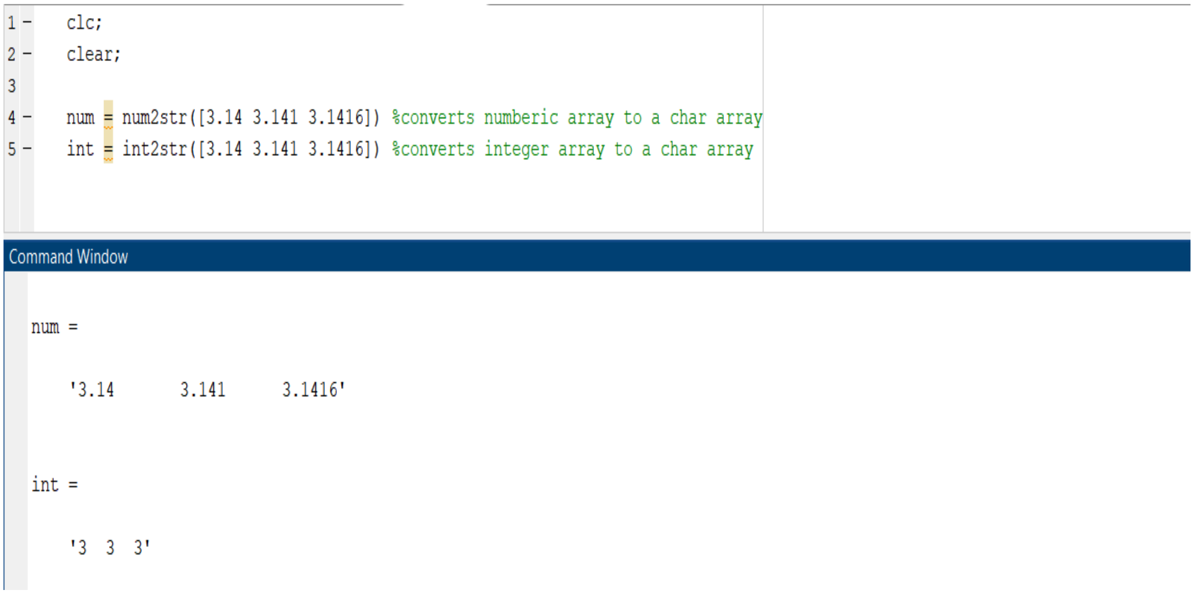
```
clc;
clear;

disp(strrep('associate professor','associate', 'assistant')); %replace 'associate' with 'assistant' in the string and disp

disp(findstr('run run run!', 'run')); %displays the starting index of each 'run' occurrence
```

```
assistant professor
     1     5     9
```

1-5 : Number to string conversion:

Name	Syntax	Description
num2str	s = num2str(A)	This function 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.
	num2str(A, precision)	This function returns a character array that represents the numbers with the maximum number of significant digits specified by precision.
	num2str(A, formatSpec)	This function pplies a format specified by formatSpec to all elements of A.

Name	Syntax	Description
int2str	K = findstr(str1, str2)	This function 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.



```
clc;
clear;

num = num2str([3.14 3.141 3.1416]) %converts numberic array to a char array
int = int2str([3.14 3.141 3.1416]) %converts integer array to a char array
```

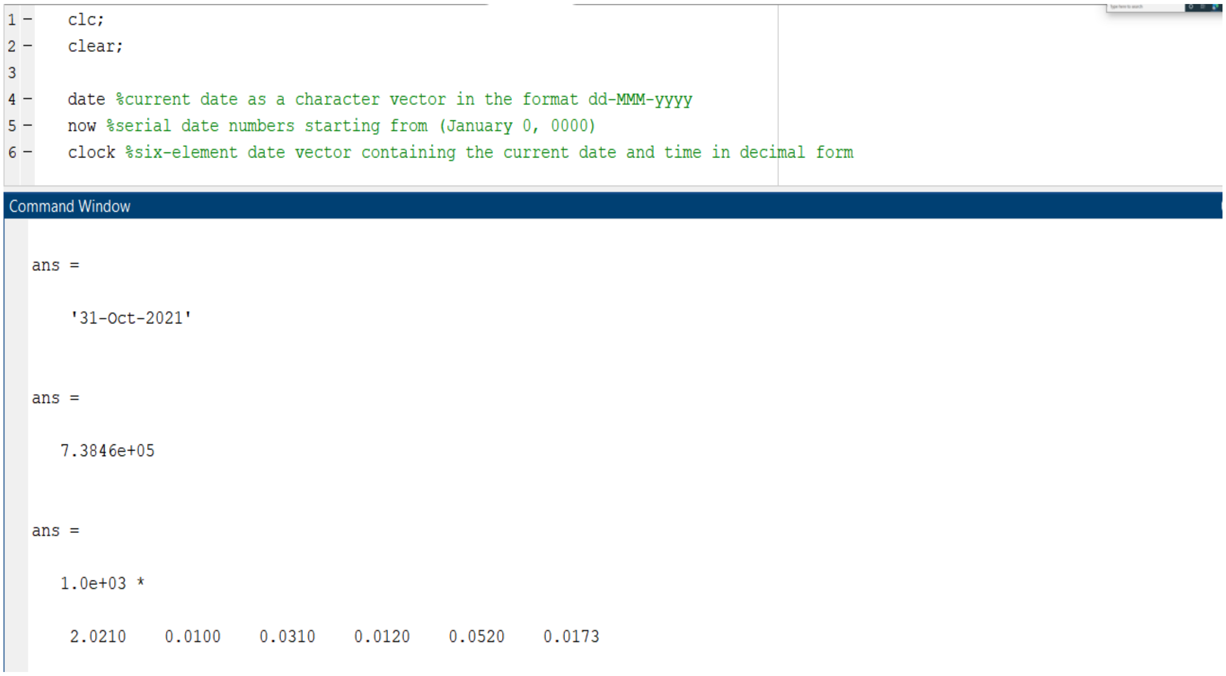
```
num =

    '3.14      3.141      3.1416'


int =

    '3  3  3'
```

1-6 : Date and time:

Name	Syntax	Description
date	c = date	This function returns the current date as a character vector in the format dd-MMM-yyyy. This format represents the day of the month (dd) as a number, the month name (MMM) as its three-letter abbreviation, and the year (yyyy) as a number.

Name	Syntax	Description
now	c = now	This function returns the current date and time as a serial date number. A serial date number represents the whole and fractional number of days starting from a fixed, preset date (January 0, 0000).

Name	Syntax	Description
clock	c = clock	This function returns a six-element date vector containing the current date and time in decimal form: [year month day hour minute seconds]
	[c tf] = clock	This function 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.

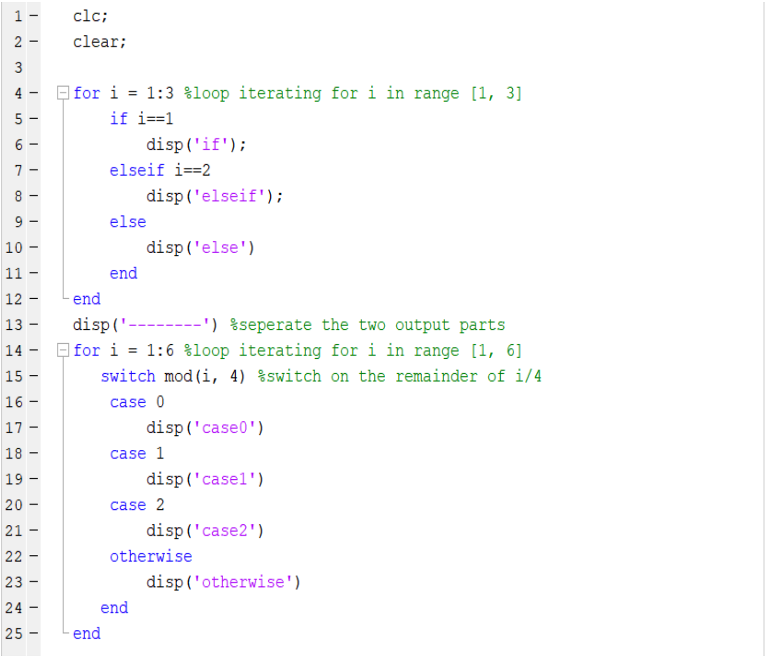


```
clc;
clear;

date %current date as a character vector in the format dd-MMM-yyyy
now %serial date numbers starting from (January 0, 0000)
clock %six-element date vector containing the current date and time in decimal form
```

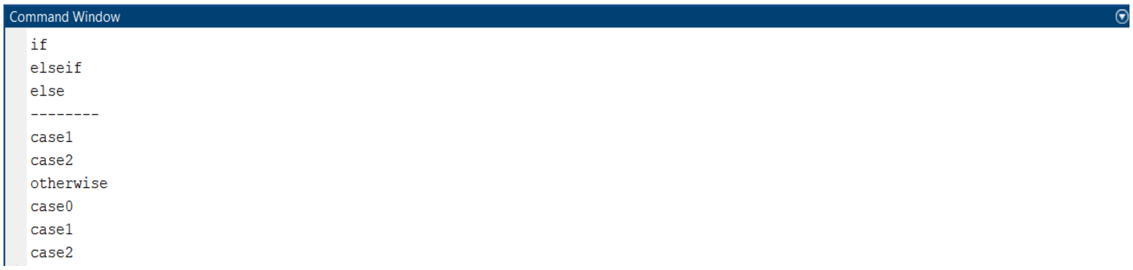
```
ans =

    '31-Oct-2021'


ans =

   7.3846e+05


ans =

   1.0e+03 *

    2.0210    0.0100    0.0310    0.0120    0.0520    0.0173
```

1-7 : Conditions:

Name	Syntax	Description
if, elseif, else	*If expression* *statements* elseif *expression* *statements* else *statements* end	This structure 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.
switch, case, otherwise	*Switch switch_expression* case *case_expression* *statements* case *case_expression* *statements* ... otherwise *statements* end	This structure 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. 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.



```
clc;
clear;

for i = 1:3 %loop iterating for i in range [1, 3]
    if i==1
        disp('if');
    elseif i==2
        disp('elseif');
    else
        disp('else')
    end
end
disp('--------') %seperate the two output parts
for i = 1:6 %loop iterating for i in range [1, 6]
   switch mod(i, 4) %switch on the remainder of i/4
    case 0
        disp('case0')
    case 1
        disp('case1')
    case 2
        disp('case2')
    otherwise
        disp('otherwise')
   end
end
```

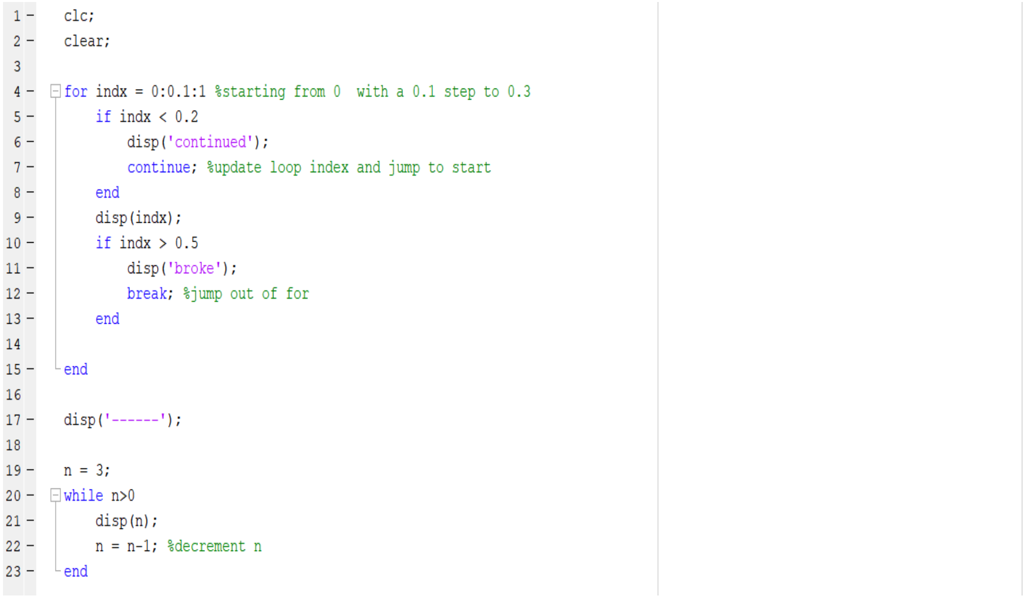


```
if
elseif
else
--------
case1
case2
otherwise
case0
case1
case2
```

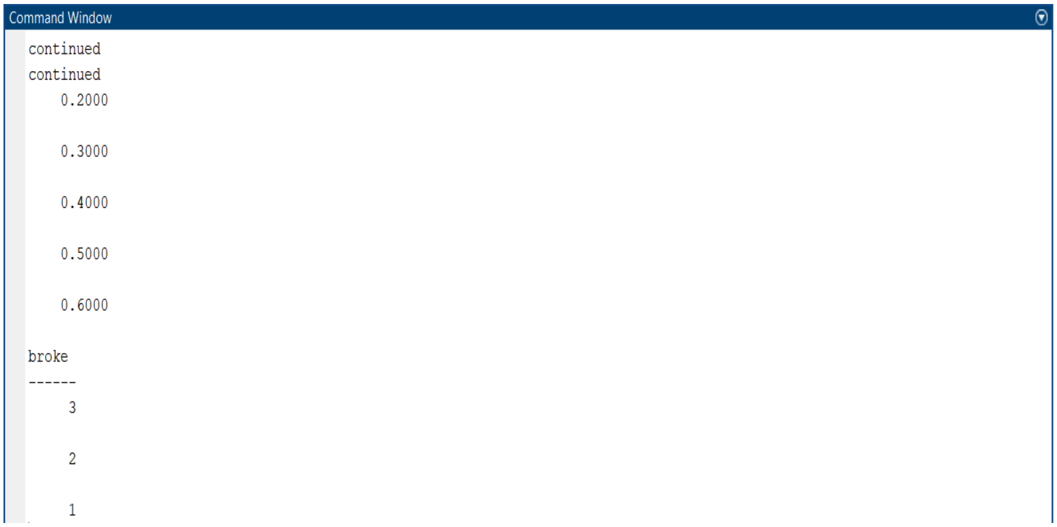
1-8 : Loops:

Name	Syntax	Description
for	*For index = values* *statements* end	*This structure* executes a group of statements in a loop for a specified number of times.

while	*while expression*
	statements
	end
		This structure 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;

for indx = 0:0.1:1 %starting from 0  with a 0.1 step to 0.3
    if indx < 0.2
        disp('continued');
        continue; %update loop index and jump to start
    end
    disp(indx);
    if indx > 0.5
        disp('broke');
        break; %jump out of for
    end

end

disp('------');

n = 3;
while n>0
    disp(n);
    n = n-1; %decrement n
end
```

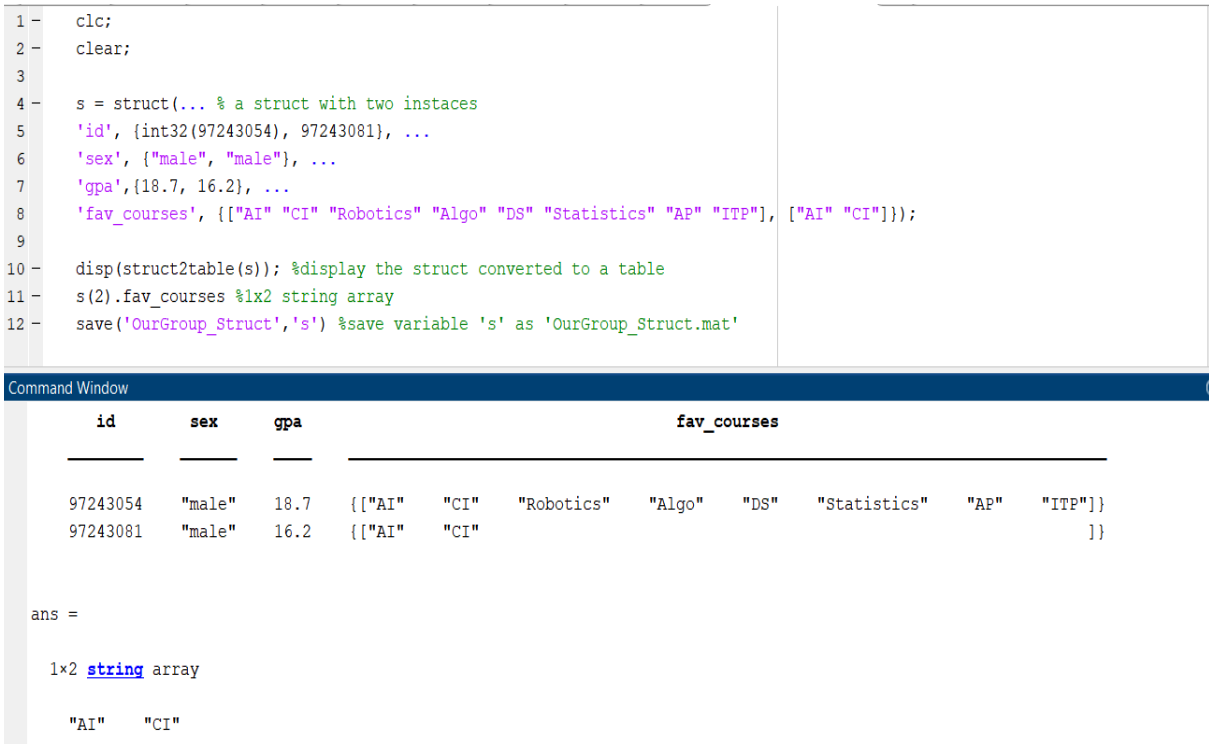


```
continued
continued
    0.2000

    0.3000

    0.4000

    0.5000

    0.6000

broke
------
     3

     2

     1
```

1-9 : Structure array:

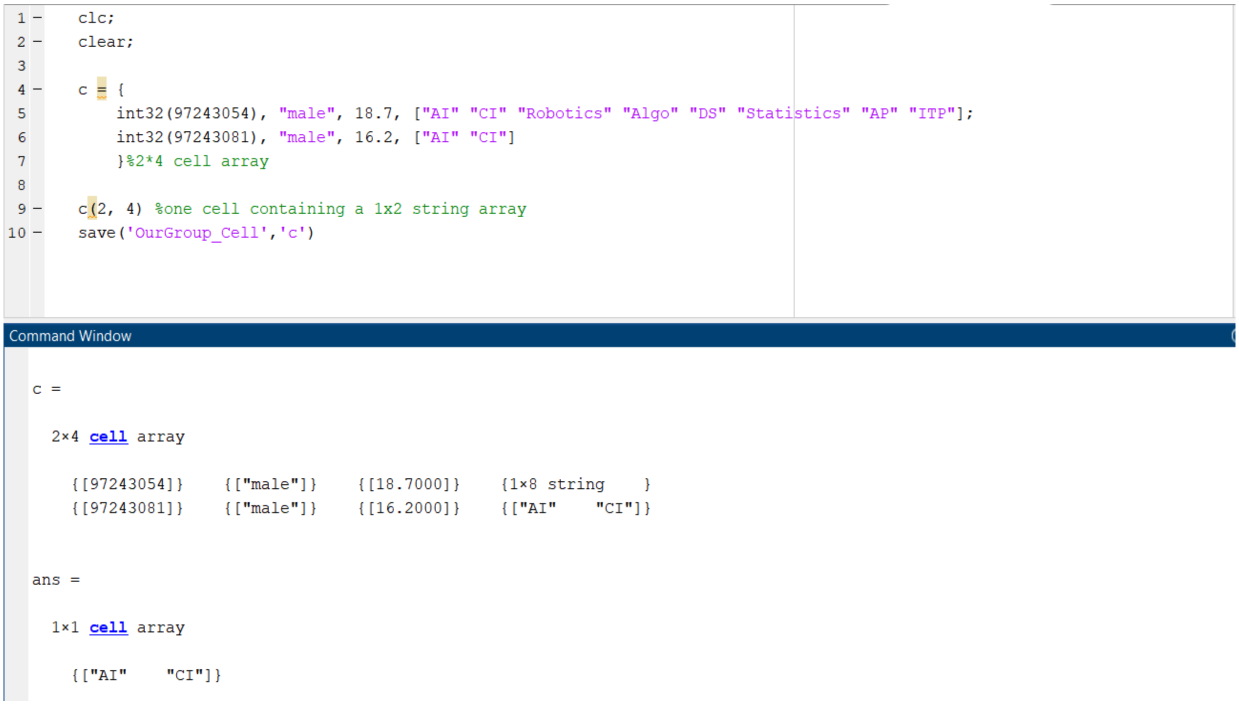
Name	Syntax	Description
structure array	s = struct	This function returns the current date and time as a serial date number. A serial date number represents the whole and fractional number of days starting from a fixed, preset date (January 0, 0000).

Name	Syntax	Description
cell array	cell	This structure is a data type with indexed data containers called cells, where each cell can contain any type of data.



```matlab
clc;
clear;

s = struct(... % a struct with two instaces
'id', {int32(97243054), 97243081}, ...
'sex', {"male", "male"}, ...
'gpa',{18.7, 16.2}, ...
'fav_courses', {["AI" "CI" "Robotics" "Algo" "DS" "Statistics" "AP" "ITP"], ["AI" "CI"]});

disp(struct2table(s)); %display the struct converted to a table
s(2).fav_courses %1x2 string array
save('OurGroup_Struct','s') %save variable 's' as 'OurGroup_Struct.mat'
```

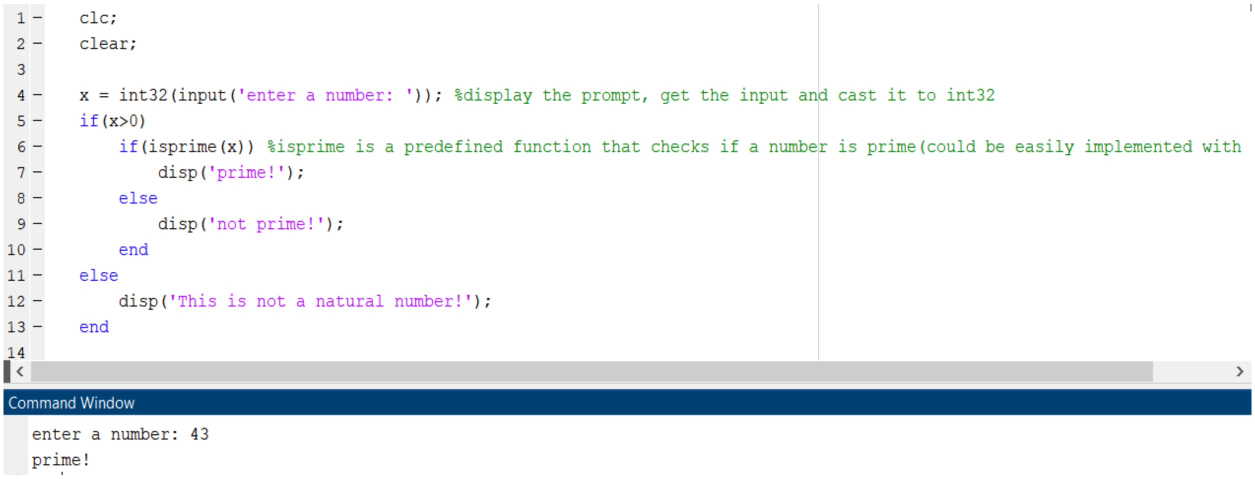
```
       id         sex       gpa                                   fav_courses
    ________    ______    ____    __________________________________________________________________________________

    97243054    "male"    18.7    {["AI"    "CI"    "Robotics"    "Algo"    "DS"    "Statistics"    "AP"    "ITP"]}
    97243081    "male"    16.2    {["AI"    "CI"                                                                  ]}


ans = 

  1×2 string array

    "AI"    "CI"
```



```matlab
clc;
clear;

c = {
    int32(97243054), "male", 18.7, ["AI" "CI" "Robotics" "Algo" "DS" "Statistics" "AP" "ITP"];
    int32(97243081), "male", 16.2, ["AI" "CI"]
    }%2*4 cell array

c(2, 4) %one cell containing a 1x2 string array
save('OurGroup_Cell','c')
```

```
c =

  2×4 cell array

    {[97243054]}    {["male"]}    {[18.7000]}    {1×8 string    }
    {[97243081]}    {["male"]}    {[16.2000]}    {["AI"    "CI"]}


ans =

  1×1 cell array

    {["AI"    "CI"]}
```

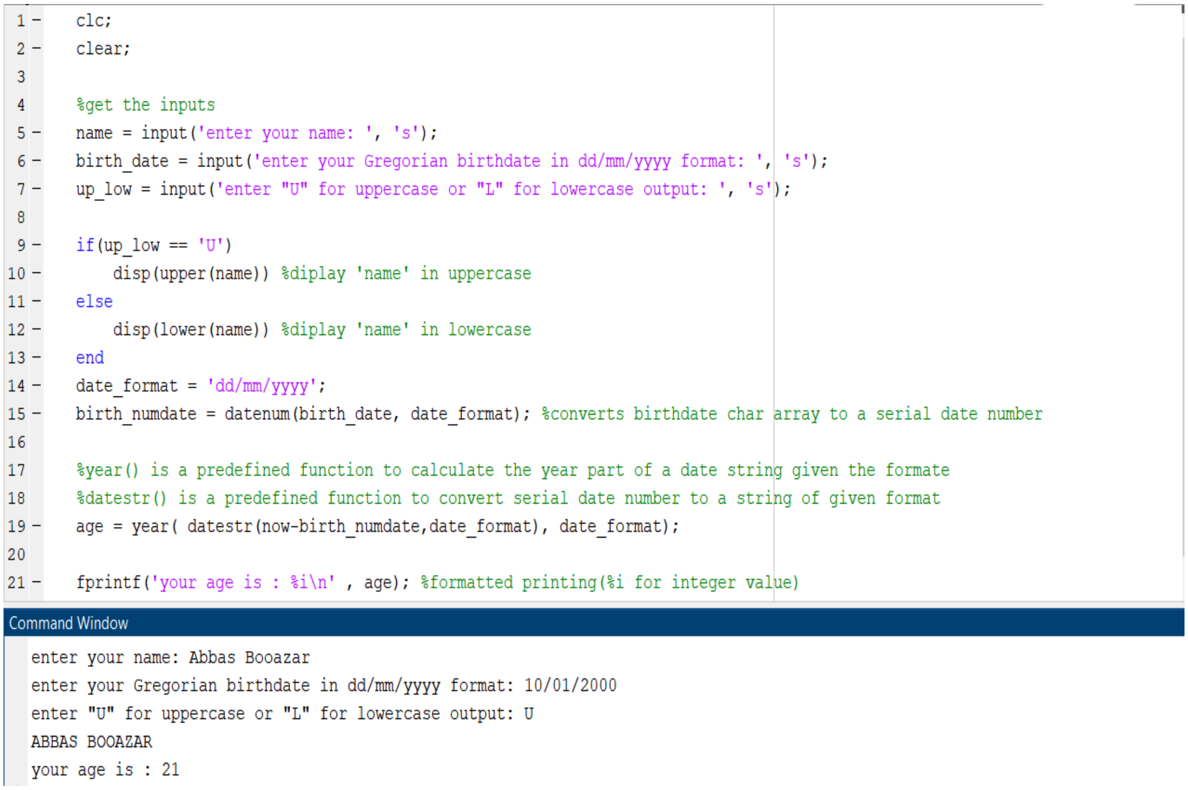
1-10:



```
clc;
clear;

x = int32(input('enter a number: ')); %display the prompt, get the input and cast it to int32
if(x>0)
    if(isprime(x)) %isprime is a predefined function that checks if a number is prime(could be easily implemented with
        disp('prime!');
    else
        disp('not prime!');
    end
else
    disp('This is not a natural number!');
end

```

```
enter a number: 43
prime!
```

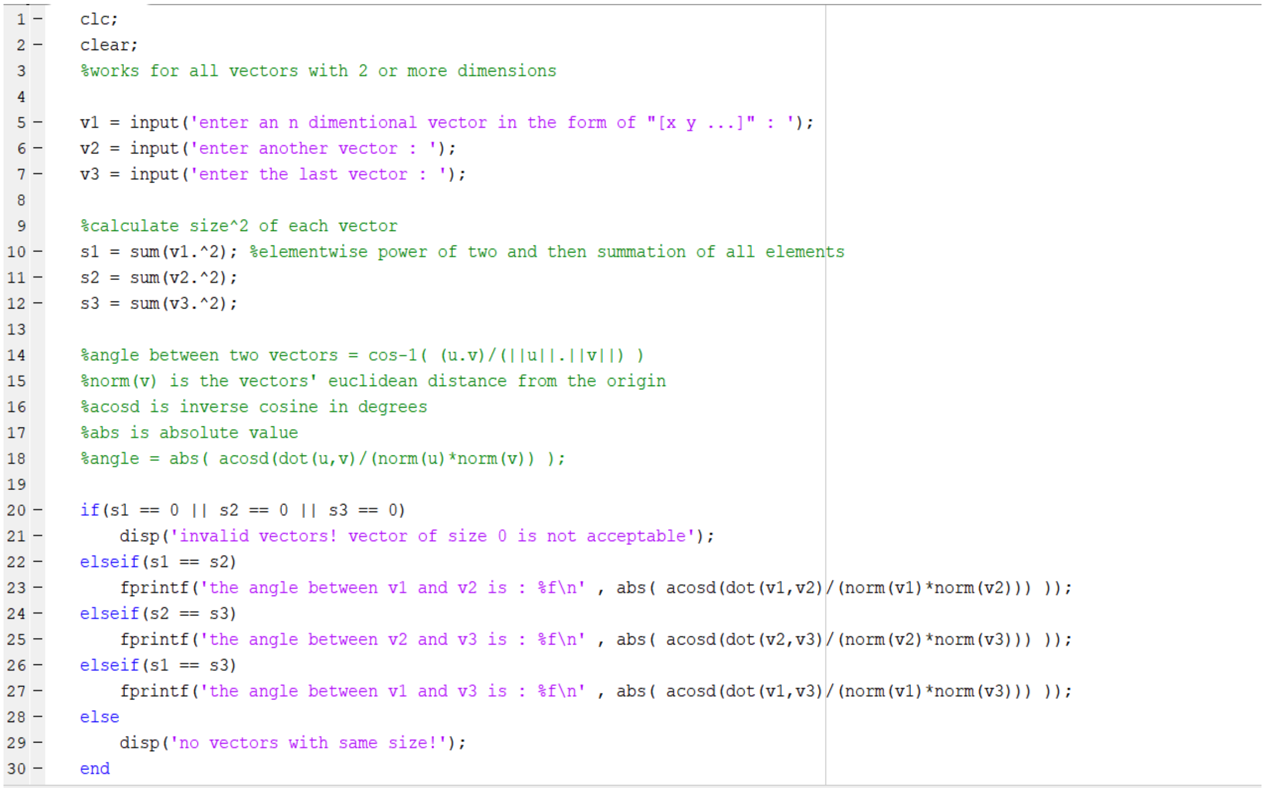
1-11:



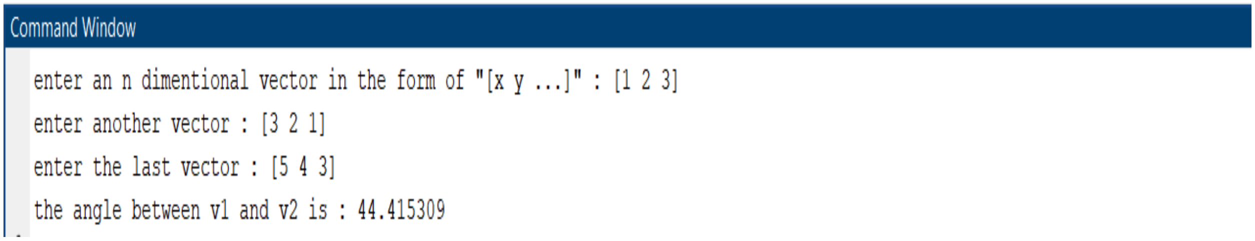
```
clc;
clear;

%get the inputs
name = input('enter your name: ', 's');
birth_date = input('enter your Gregorian birthdate in dd/mm/yyyy format: ', 's');
up_low = input('enter "U" for uppercase or "L" for lowercase output: ', 's');

if(up_low == 'U')
    disp(upper(name)) %diplay 'name' in uppercase
else
    disp(lower(name)) %diplay 'name' in lowercase
end
date_format = 'dd/mm/yyyy';
birth_numdate = datenum(birth_date, date_format); %converts birthdate char array to a serial date number

%year() is a predefined function to calculate the year part of a date string given the formate
%datestr() is a predefined function to convert serial date number to a string of given format
age = year( datestr(now-birth_numdate,date_format), date_format);

fprintf('your age is : %i\n' , age); %formatted printing(%i for integer value)
```

```
enter your name: Abbas Booazar
enter your Gregorian birthdate in dd/mm/yyyy format: 10/01/2000
enter "U" for uppercase or "L" for lowercase output: U
ABBAS BOOAZAR
your age is : 21
```

1-12:



```
clc;
clear;
%works for all vectors with 2 or more dimensions

v1 = input('enter an n dimentional vector in the form of "[x y ...]" : ');
v2 = input('enter another vector : ');
v3 = input('enter the last vector : ');

%calculate size^2 of each vector
s1 = sum(v1.^2); %elementwise power of two and then summation of all elements
s2 = sum(v2.^2);
s3 = sum(v3.^2);

%angle between two vectors = cos-1( (u.v)/(||u||.||v||) )
%norm(v) is the vectors' euclidean distance from the origin
%acosd is inverse cosine in degrees
%abs is absolute value
%angle = abs( acosd(dot(u,v)/(norm(u)*norm(v)) );

if(s1 == 0 || s2 == 0 || s3 == 0)
    disp('invalid vectors! vector of size 0 is not acceptable');
elseif(s1 == s2)
    fprintf('the angle between v1 and v2 is : %f\n' , abs( acosd(dot(v1,v2)/(norm(v1)*norm(v2))) ));
elseif(s2 == s3)
    fprintf('the angle between v2 and v3 is : %f\n' , abs( acosd(dot(v2,v3)/(norm(v2)*norm(v3))) ));
elseif(s1 == s3)
    fprintf('the angle between v1 and v3 is : %f\n' , abs( acosd(dot(v1,v3)/(norm(v1)*norm(v3))) ));
else
    disp('no vectors with same size!');
end
```



```
enter an n dimentional vector in the form of "[x y ...]" : [1 2 3]
enter another vector : [3 2 1]
enter the last vector : [5 4 3]
the angle between v1 and v2 is : 44.415309
```