DEPARTMENT OF ELECTRICAL ENGINEERING
SUKKUR IBA University
Signals and Systems (Lab)

Name: Muhammad Shayan Date: 18/09/2023
Score: ______________________Signature of Instructor___________________________________

Lab Practice # 05
MATLAB PROGRAMMING (Control flow)

Objective:

To understand following concepts using Matlab programs

- Conditional Control — if, else, switch
- Loop Control — for, while, continue, break
- Vectorization
- Preallocation

Theory

Conditional Control — if, else, switch

Conditional statements enable you to select at run time which block of code to execute. The simplest conditional statement is an if statement. For example:

```
% use of if
% Generate a random number 1~100
a = randi(100,1)
% If it is even, divide by 2
if rem(a, 2) == 0
disp('a is even')
end
```

if statements can include alternate choices, using the optional keywords elseif or else. For example:

```
% Use of if else
% Generate a random number
a = randi(100,1)
if rem(a, 2) == 0
disp('a is even')
else
disp('a is odd')
end


% Use of else if
a = randi(100, 1)
if a < 30
disp('number is smaller than 30')
elseif a < 80
disp('number is in between 29 and 80')
else
disp('number is greater than 79 and less than 101')
end


%Enter number as an input from user
a = input('Enter a number: ');
if a < 30
disp('number is smaller than 30')
elseif a < 80
```

```
disp('number is in between 29 and 80')
else
disp('number is greater than 79')
end
```

Alternatively, when you want to test for equality against a set of known values, use a switch statement. For example:

```
% switch example with the use of function "weekday and date".
%[dayNum, dayString] = weekday('18-Feb-2019');
% fixed date
%[dayNum, dayString] = weekday(date,'long','en_US');
% Automatic date with long "Monday" or short "Mon"
[dayNum, dayString] = weekday(date,'long','local');


switch dayString
case 'Mon'
disp('Mon is short for monday')
case 'Monday'
disp('Start of the work week')
case 'Tuesday'
disp('Day 2')
case 'Wednesday'
disp('Day 3')
case 'Thursday'
disp('Day 4')
case 'Friday'
disp('Last day of the work week')
otherwise
disp('Weekend!')
end


%Same switch example with user input as a number from 1~7
disp('We have 7 cases, one for each day')
a=input('Enter number = ')
switch a
case 1
disp('Start of the work week')
case 2
disp('Day 2')
case 3
disp('Day 3')
case 4
disp('Day 4')
case 5
disp('Last day of the work week')
otherwise
disp('Weekend!')
end


%Same switch example with user input as a character from a~d
disp('Select any of four cases from a, b, c or d')
abcd=input('Enter a character form a to d = ','s');
switch abcd
case 'aa'
disp('Character a')
case 'b'
disp('Character b')
case 'c'
disp('Character c')
case 'd'
disp('Character d')
otherwise
```

```
disp('other than a, b, c, d is pressed')
end
```

For both if and switch, MATLAB executes the code corresponding to the first true condition, and then exits the code block. Each conditional statement requires the end keyword. In general, when you have many possible discrete, known values, switch statements are easier to read than if statements. However, you cannot test for inequality between switch and case values. For example, you cannot implement this type of condition with a switch:

```
%% Can't use switch in situation where there are limited outputs but too many
cases.
yourNumber = input('Enter a number: ');
if yourNumber < 0
disp('Negative')
elseif yourNumber > 0
disp('Positive')
else
disp('Zero')
end
```

Array Comparisons in Conditional Statements
It is important to understand how relational operators and if statements work with matrices. When you want to check for equality between two variables, you might use if A == B, ... This is valid MATLAB code, and does what you expect when A and B are scalars. But when A and B are matrices, A == B does not test *if* they are equal, it tests *where* they are equal; the result is another matrix of 0s and 1s showing element-by-element equality.

```
% Element by element comparison
A = magic(3)
% 3x3 matrix with sum of elements in either row/column/diagonal is 15
B = A;
B(1,1) = 0
%just first element of B is 0
A == B
%returns a matrix with all elements 1, expect first element

%% use of "isequal" Logical0
A = magic(4)
B = A;
B(1,1) = 0
isequal(A,B)

%% use of "isequal" Logical 1
A = magic(4)
B = A;
isequal(A,B)
```

The proper way to check for equality between two variables is to use the isequal function: if isequal(A, B), ... isequal returns a *scalar* logical value of 1 (representing true) or 0 (false), instead of a matrix, as the expression to be evaluated by the if function.

Here is another example to emphasize this point. If A and B are scalars, the following program will never reach the “unexpected situation”. But for most pairs of matrices, including our magic squares with interchanged columns, none of the matrix conditions A > B, A < B, or A == B is true for *all* elements and so the else clause is executed:

```
%% Comparison + Conditional statement
A=input('Input value for A = ')
% can also compare matrices
B=input('Input value for B = ')
if A > B
'A is greater than B'
elseif A < B
'A is less than B'
elseif A == B
disp('A and B are equal')
else
error('Unexpected situation')
```

```
end
```

Several functions are helpful for reducing the results of matrix comparisons to scalar conditions for use with if, including

isequal
isempty
all
any

```
%% it's empty, output will be "logical 1"
A=[];
isempty(A)

%% it's not empty, output will be "logical 0"
A=[0];
isempty(A)

%% if all elements are non-zero then returns logical 1 else logical 0
A=[1 1 0];
all(A)

%% if any element is non-zero then returns logical 1 else logical 0
A=[1 1 1];
any(A)
```

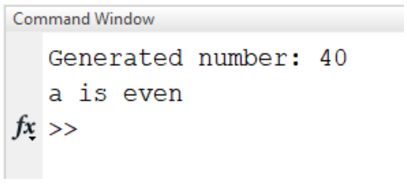
Procedure

Type these and save them with, and then run them.

Observation

Run above programs and comment on the program structure and output.

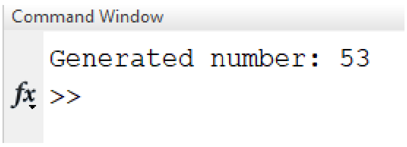
Program – 1:
Output:



Command Window

```
Generated number: 40
a is even
>>
```

For Even



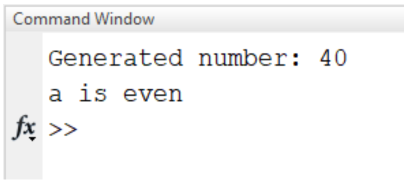
Command Window

```
Generated number: 53
>>
```

For Odd

Comment: Tells if the randomly generated number is even.

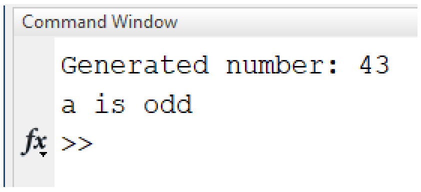
Program – 2:
Output:



Command Window

```
Generated number: 40
a is even
>>
```

For Even



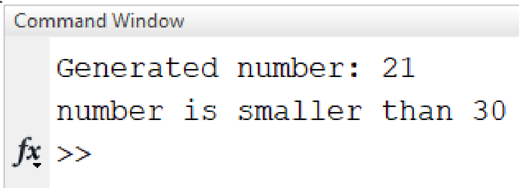
Command Window

```
Generated number: 43
a is odd
>>
```

For Odd

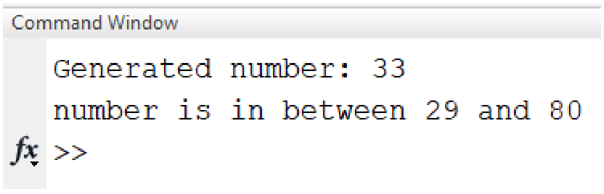
Comment: Tells if the randomly generated number is even or odd.

Program – 3:
Output:



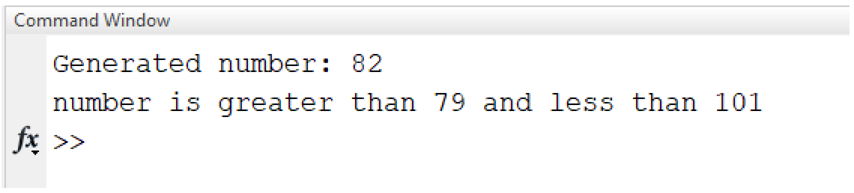
Command Window

```
Generated number: 21
number is smaller than 30
>>
```



Command Window

```
Generated number: 33
number is in between 29 and 80
>>
```

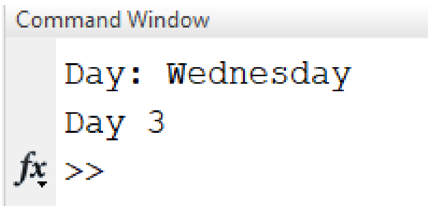


Command Window

```
Generated number: 82
number is greater than 79 and less than 101
>>
```

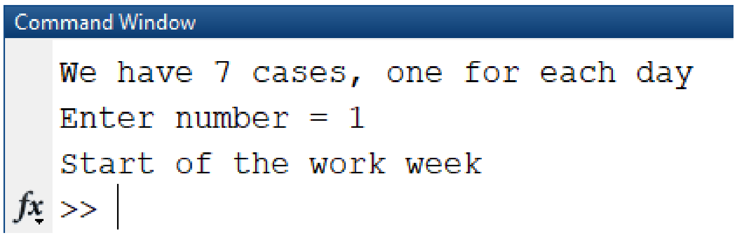
Comment: Tells that the randomly generated number lies in which user defined limit.

Program – 4:
Output:


```
Day: Wednesday
Day 3
>>
```

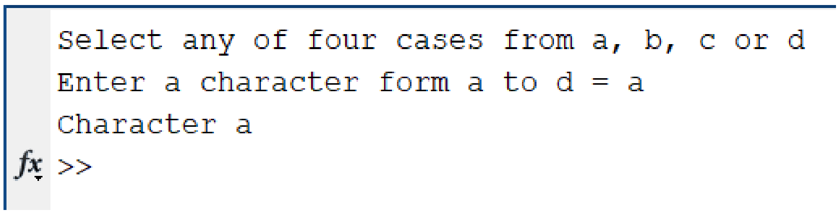
Comment: Accesses the local **Date and Time** and displays the current day.

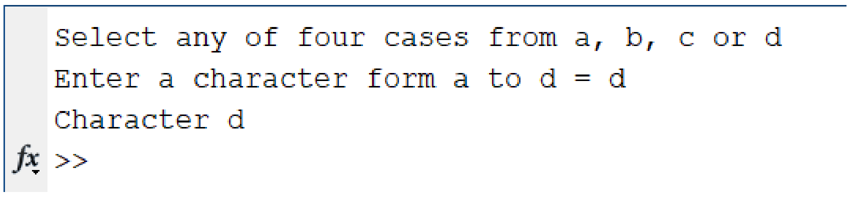
Program – 5:
Output:

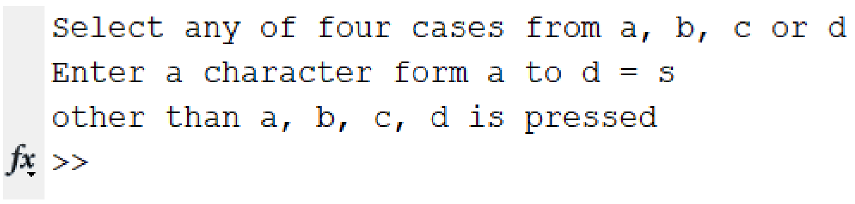

```
We have 7 cases, one for each day
Enter number = 1
Start of the work week
>>
```

Comment: Asks the user to input a day number, then shows its case corresponding definition.

Program – 6:
Output:

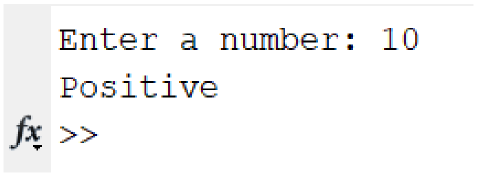

```
Select any of four cases from a, b, c or d
Enter a character form a to d = a
Character a
>>
```


```
Select any of four cases from a, b, c or d
Enter a character form a to d = d
Character d
>>
```


```
Select any of four cases from a, b, c or d
Enter a character form a to d = s
other than a, b, c, d is pressed
>>
```

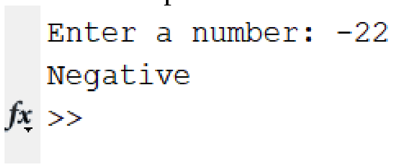
Comment: Asks the user to input a character, then uses the switch and tells if something matches from the cases.

Program – 7:
Output:



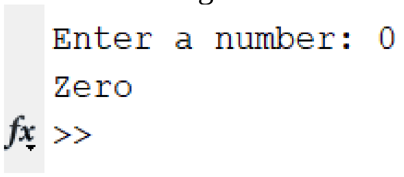
```
Enter a number: 10
Positive
>>
```

for positive number



```
Enter a number: -22
Negative
>>
```

For negative number

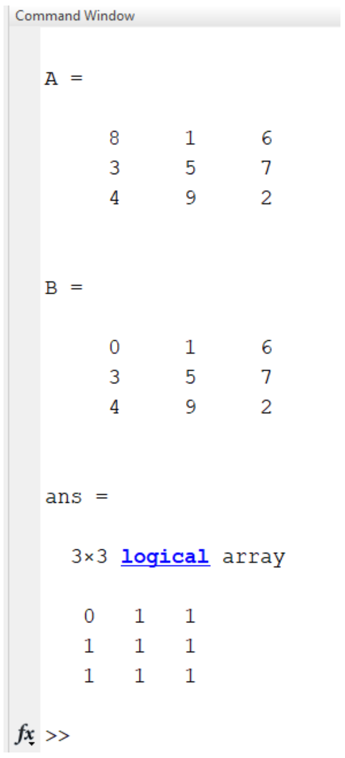


```
Enter a number: 0
Zero
>>
```

for zero

Comment: Tells whether the input number is positive, negative, or zero.

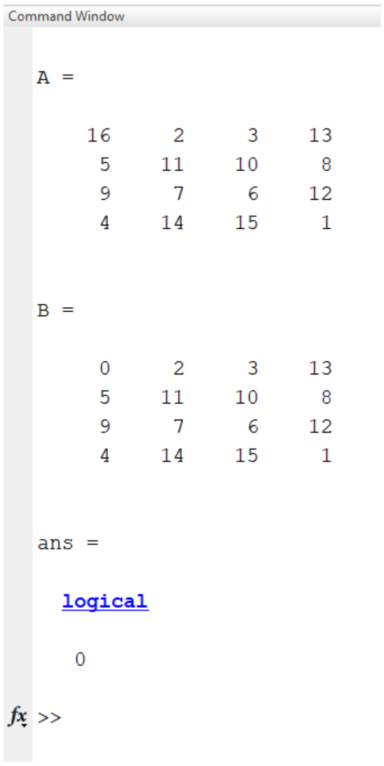
Program – 8:
Output:



```
A =

     8     1     6
     3     5     7
     4     9     2


B =

     0     1     6
     3     5     7
     4     9     2


ans =

  3×3 logical array

   0   1   1
   1   1   1
   1   1   1

>>
```

Comment: Firstly, a 3x3 magic matrix A is generated, then a copy of A is created inside matrix B, then (1,1) of B has been replaced by 0; then both matrices are compared element by element. 1 for matched, and 0 for unmatched.

Program – 9:
Output:

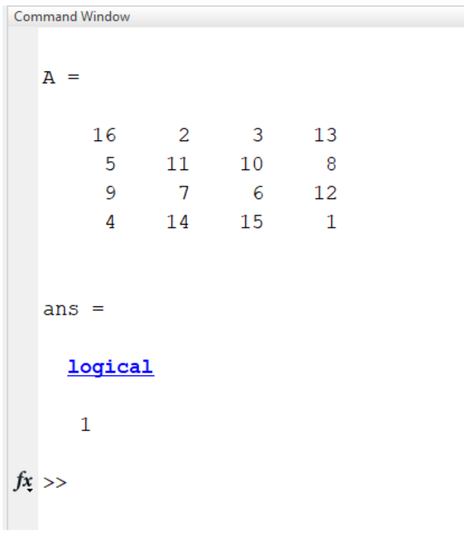


```
A =

    16     2     3    13
     5    11    10     8
     9     7     6    12
     4    14    15     1


B =

     0     2     3    13
     5    11    10     8
     9     7     6    12
     4    14    15     1


ans =

  logical

   0

fx >>
```

Comment: Firstly, a 4x4 magic matrix A is generated, then a copy of A is created inside matrix B, then (1,1) of B has been replaced by 0; then both matrices are compared using **isequal()** function it returns logical 0 as both matrices are not equal due to change in matrix B.

.

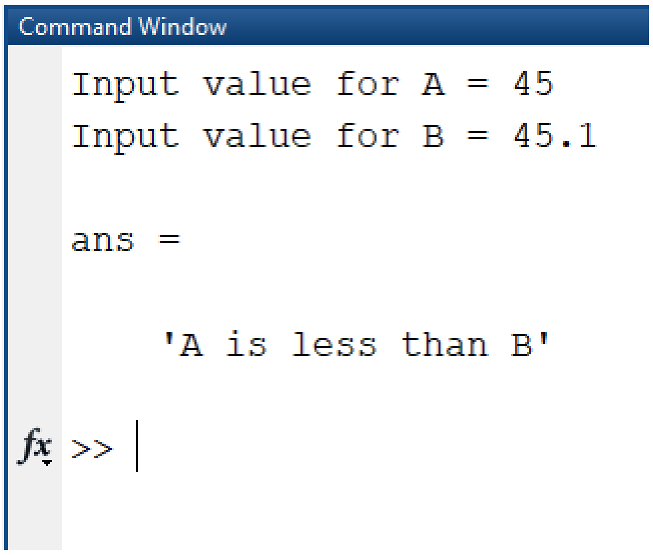
Program – 10:
Output:



```
A =

    16     2     3    13
     5    11    10     8
     9     7     6    12
     4    14    15     1


ans =

  logical

   1

fx >>
```

Comment: Firstly, a 4x4 magic matrix A is generated, then a copy of A is created inside matrix B, then both matrices are compared using **isequal()** function it returns logical 1 as both matrix B is clone of matrix A.

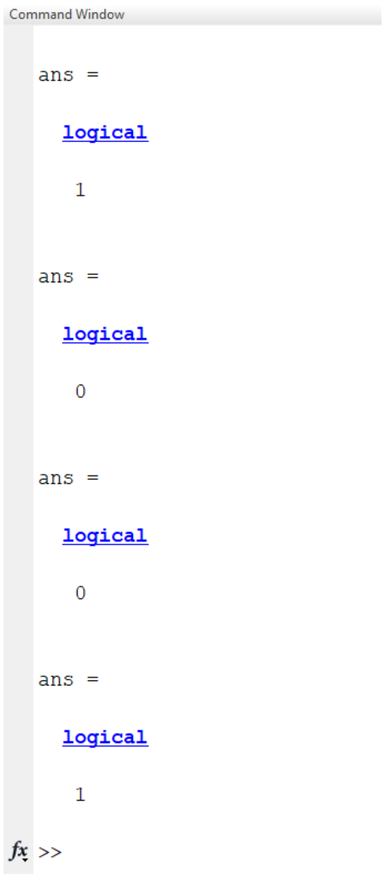
Program – 11:
Output:



```
Input value for A = 45
Input value for B = 45.1

ans =

    'A is less than B'

>>
```

Comment: This program takes two input values for variables A and B then compares both the values and tells if A is lesser or greater than, or equal compared to B.

Program – 12:
Output:



```
ans =

  logical

   1


ans =

  logical

   0


ans =

  logical

   0


ans =

  logical

   1

>>
```

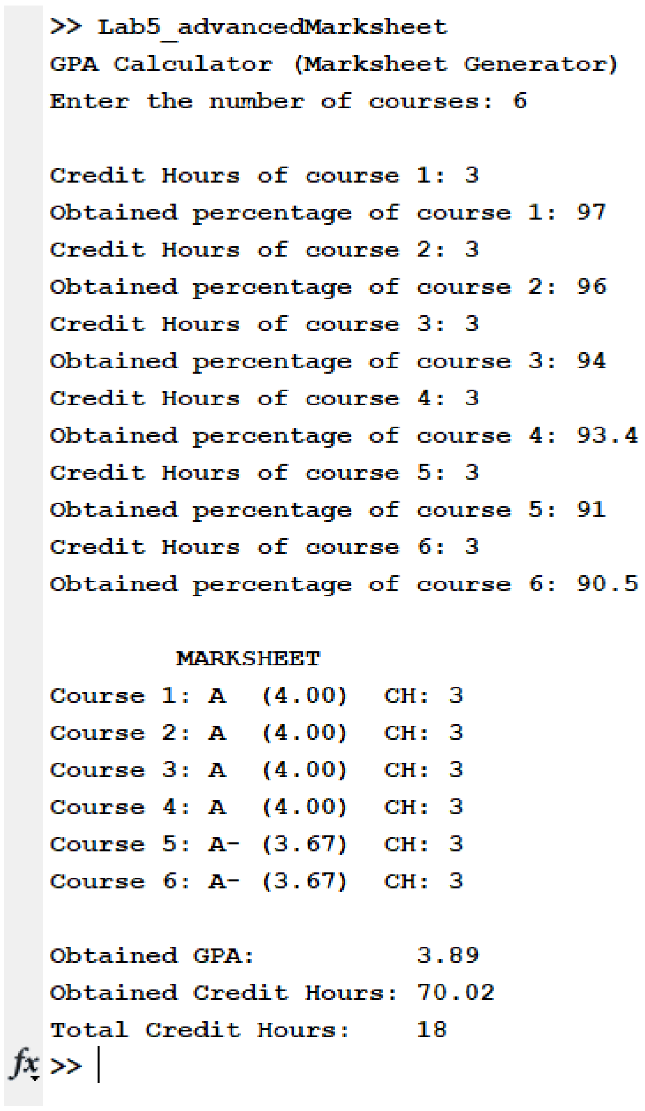
Comment: **isempty()** returns 1 only when there is nothing, not even a zero, otherwise it returns 0. **all()** returns 1 when all values are non-zero, otherwise returns 0, while **any()** returns 1 if even a single non-zero value is present.

Exercise

Make advanced level marksheet program that takes data of student and calculates its grade and finally show result like transcript.

*"Click here to download **m** file"*

Result:



```
>> Lab5_advancedMarksheet
GPA Calculator (Marksheet Generator)
Enter the number of courses: 6

Credit Hours of course 1: 3
Obtained percentage of course 1: 97
Credit Hours of course 2: 3
Obtained percentage of course 2: 96
Credit Hours of course 3: 3
Obtained percentage of course 3: 94
Credit Hours of course 4: 3
Obtained percentage of course 4: 93.4
Credit Hours of course 5: 3
Obtained percentage of course 5: 91
Credit Hours of course 6: 3
Obtained percentage of course 6: 90.5

        MARKSHEET
Course 1: A  (4.00)  CH: 3
Course 2: A  (4.00)  CH: 3
Course 3: A  (4.00)  CH: 3
Course 4: A  (4.00)  CH: 3
Course 5: A- (3.67)  CH: 3
Course 6: A- (3.67)  CH: 3

Obtained GPA:          3.89
Obtained Credit Hours: 70.02
Total Credit Hours:    18
fx >>
```

Review Questions

1. What are conditional controls, give their names and describe uses?

Conditional controls in MATLAB, such as if and switch statements, enables to execute code based on specific conditions. "if" statements are used for single condition branching, while "switch" statements are ideal for selecting code blocks based on predefined values, simplifying decision-making in the MATLAB programs.