**TITLE OF LAB: (GETTING STARTED WITH MATLAB)**

**LAB # 01**



**Spring 2022**

**CSE301L Signals & Systems Lab**

Submitted by: **Safi Ullah Khan**

Registration No.: **20PWCSE1943**

Class Section: **B**

“On my honor, as student of University of Engineering and Technology, I have neither given nor received unauthorized assistance on this academic work.”

Student Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Submitted to:

**Engr. Durr-e-Nayab**

Day, Date (e.g. Friday, April 22th, 2022)

**Department of Computer Systems Engineering**

**University of Engineering and Technology, Peshawar**

**OBJECTIVES OF THE LAB:**

Matlab will be used extensively in all the succeeding labs. The goal of this first lab is to gain familiarity with Matlab and build some basic skills in the Matlab language. Some specific topics covered in this lab

are:

* Introduction to Matlab
* Matlab Environment
* Matlab Help
* Variable arithmetic
* Built in Mathematical Functions
* Input and display
* Timing functions
* Introduction to M‐files

**NOTE: I DID FORFORM ALL THE TASK IN THE END OF THE MATLAB INTRODUCTIONS:**

* 1. **WHAT IS MATLAB?**

MATLAB is a high-performance language for technical computing. It integrates computation, visualization, and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation. Typical uses include:

* Math and computation
* Algorithm development
* Modeling, simulation, and prototyping
* Data analysis, exploration, and visualization
* Scientific and engineering graphics
* Application development, including Graphical User Interface building

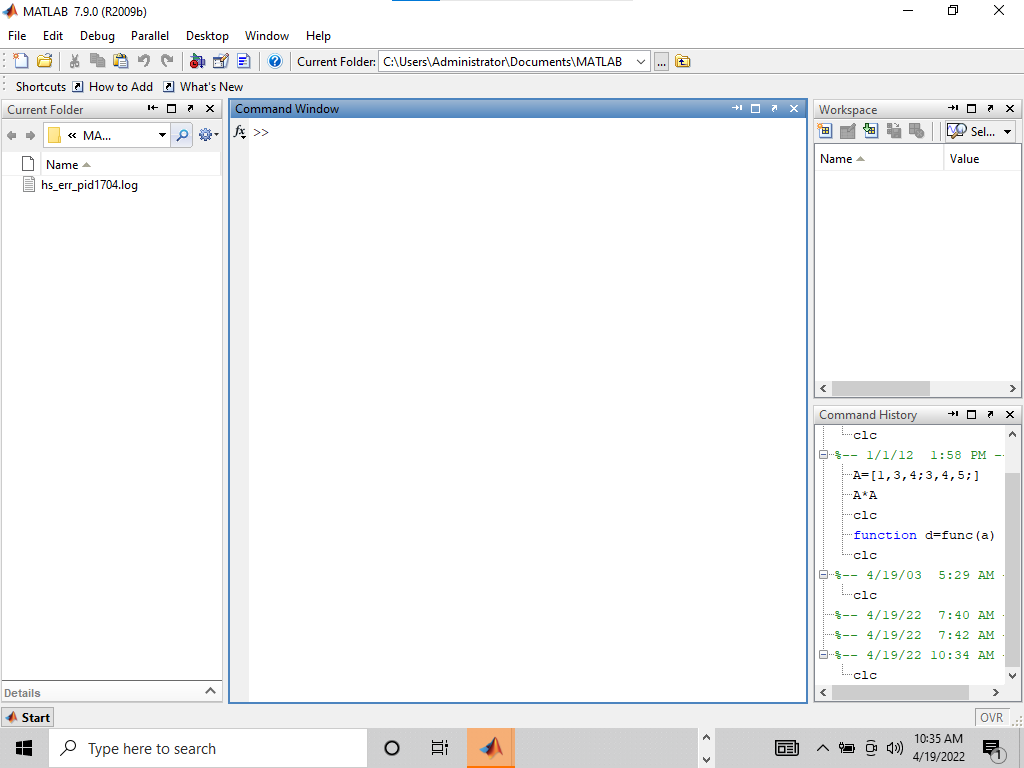
**The MATLAB System:**

* The matlab language
* the MATLAB working environment
* Handle Graphics.
* The MATLAB mathematical function library
* The MATLAB Application Program Interface (API).
  1. **ENTERING AND RUNNING MATLAB:**

Double click on the MATLAB icon to launch and a command window will appear with the prompt:

>>

You are now in MATLAB. From this point on, individual MATLAB commands may be given at the program prompt. They will be processed when you hit the <ENTER> key. The following figure shows the screenshot of matlab.



**1.3 LEAVING MATLAB:**

A MATLAB session may be terminated by simply typing

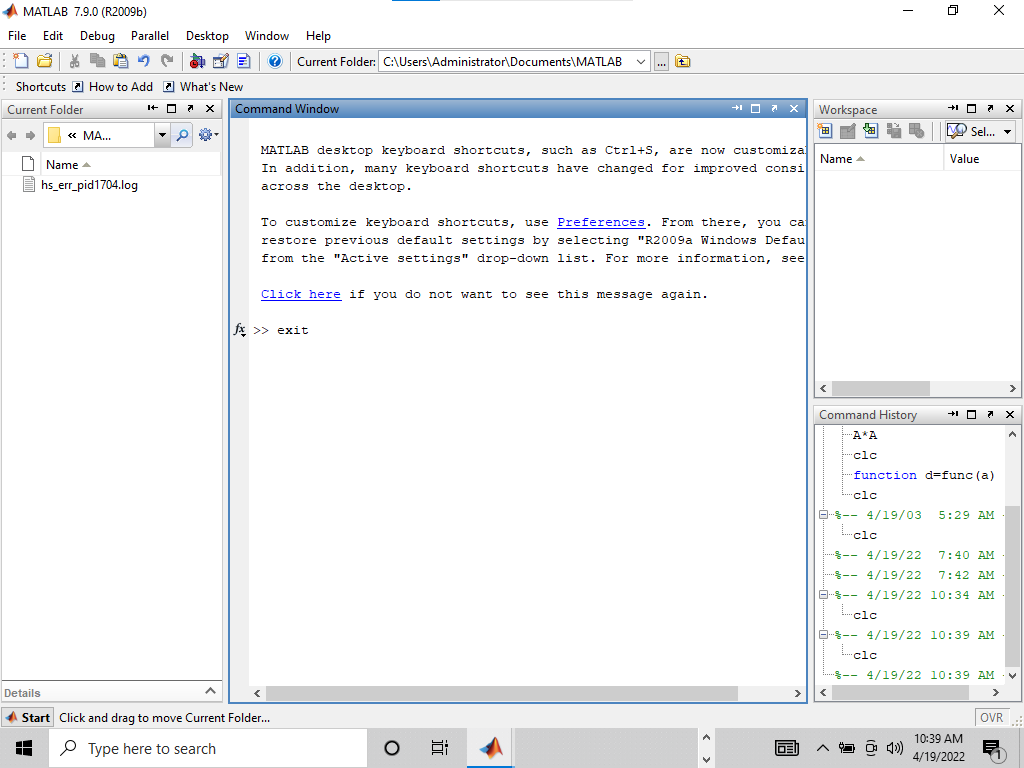
>> quit

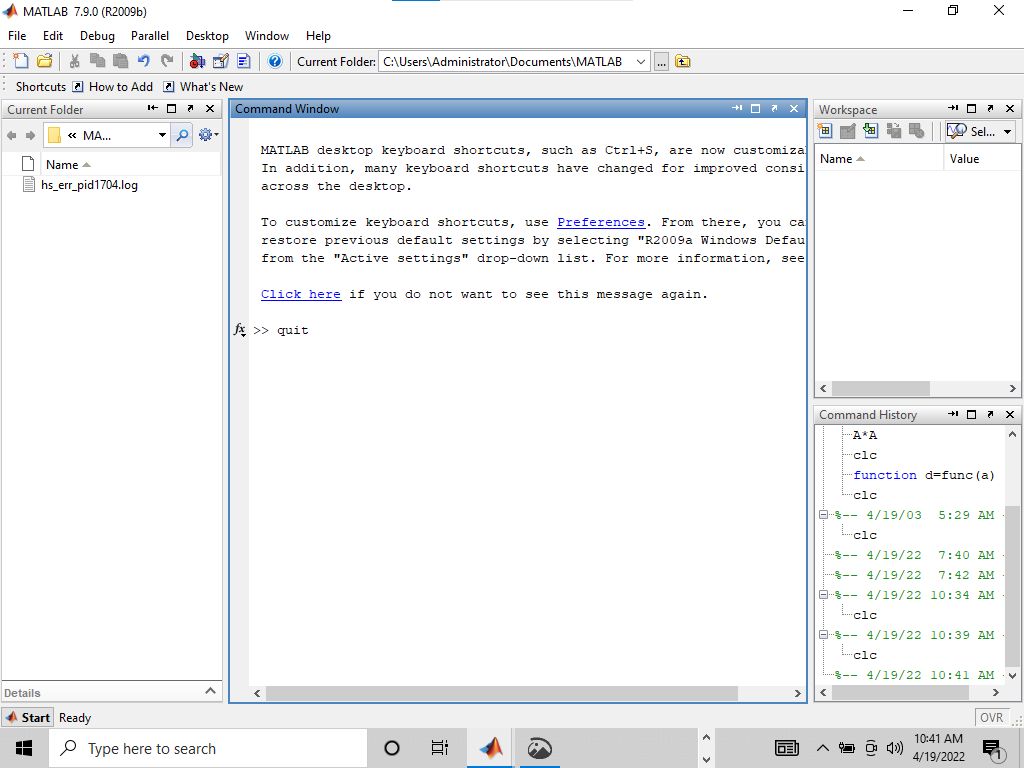
or by typing

>> exit

at the MATLAB prompt.

Screenshot:





1.4 **MATLAB HELP:**

Online help is available from the MATLAB prompt, both generally (listing all available

commands). >> help

[a long list of help topics follows]

and for specific commands:

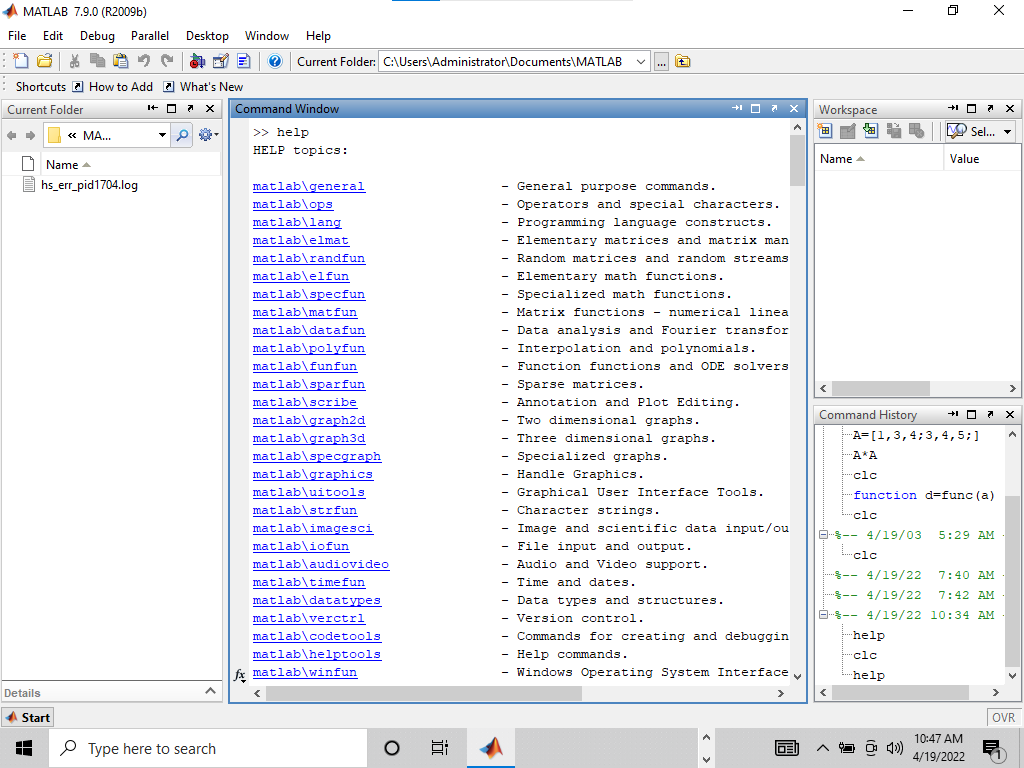
>> help command name

If you want to search for all the commands related to some particular functionality, use the keyword

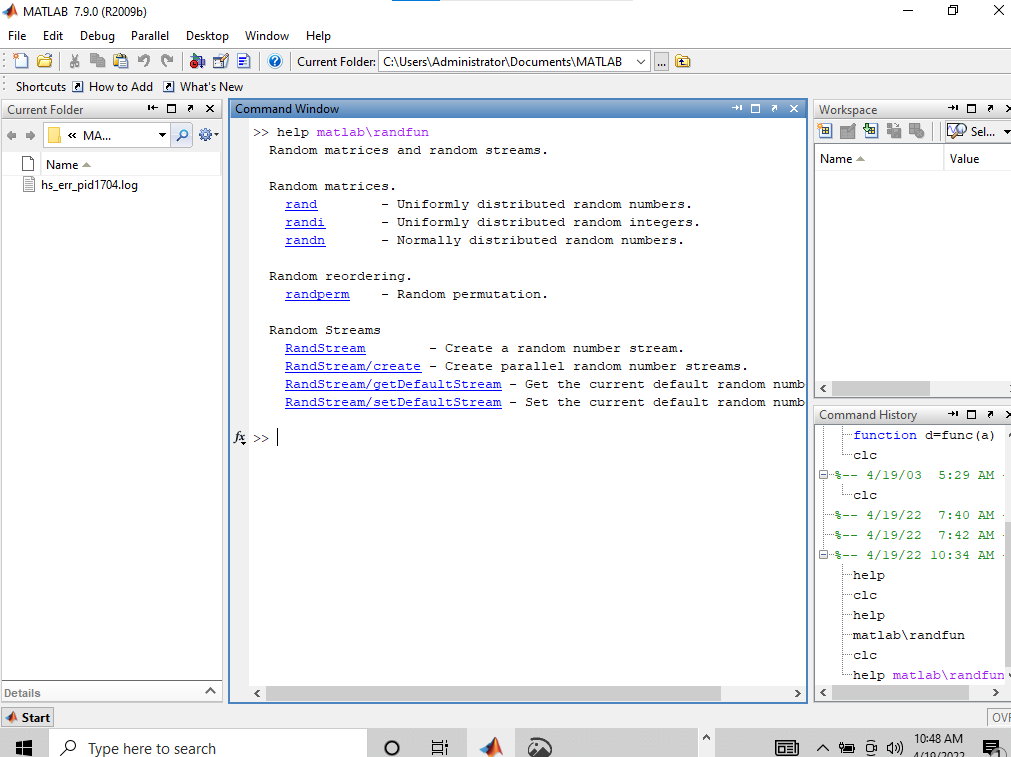
look for followed by a keyword that explains the functionality.

>> look for convolution

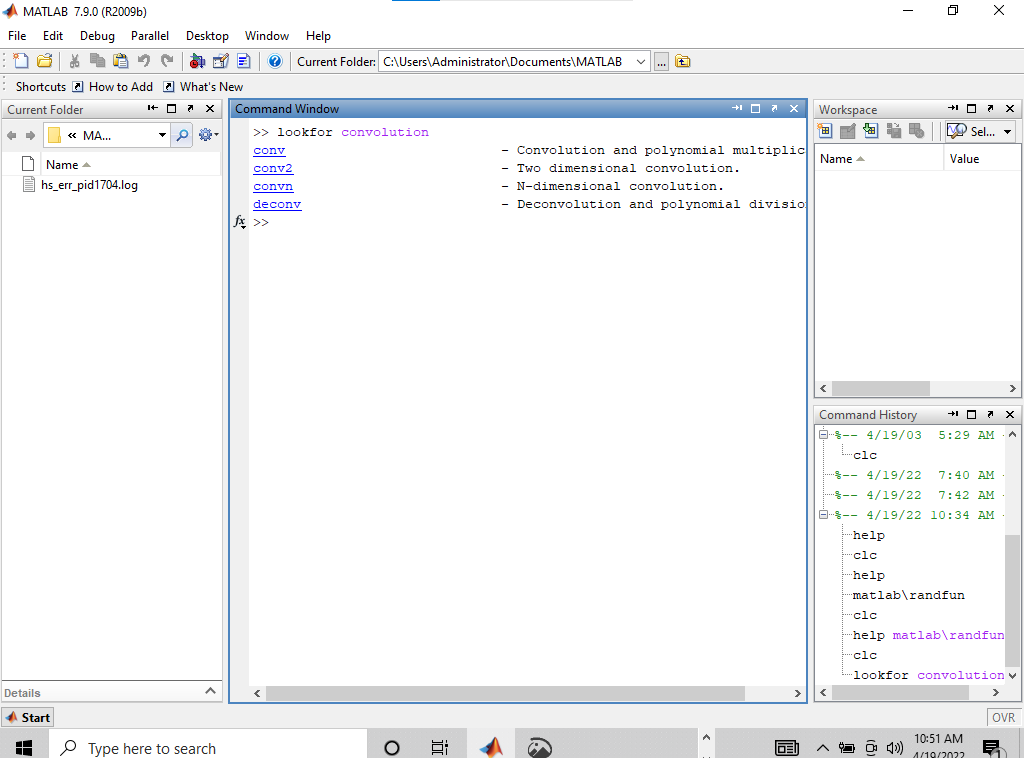
will return a number of commands that perform convolution related tasks.



and for specific commands:



>>look for convolution



**1.5 VARIABLES:**

MATLAB has built‐in variables like pi, eps, and ans. You can learn their values from

the MATLAB interpreter.

>> eps

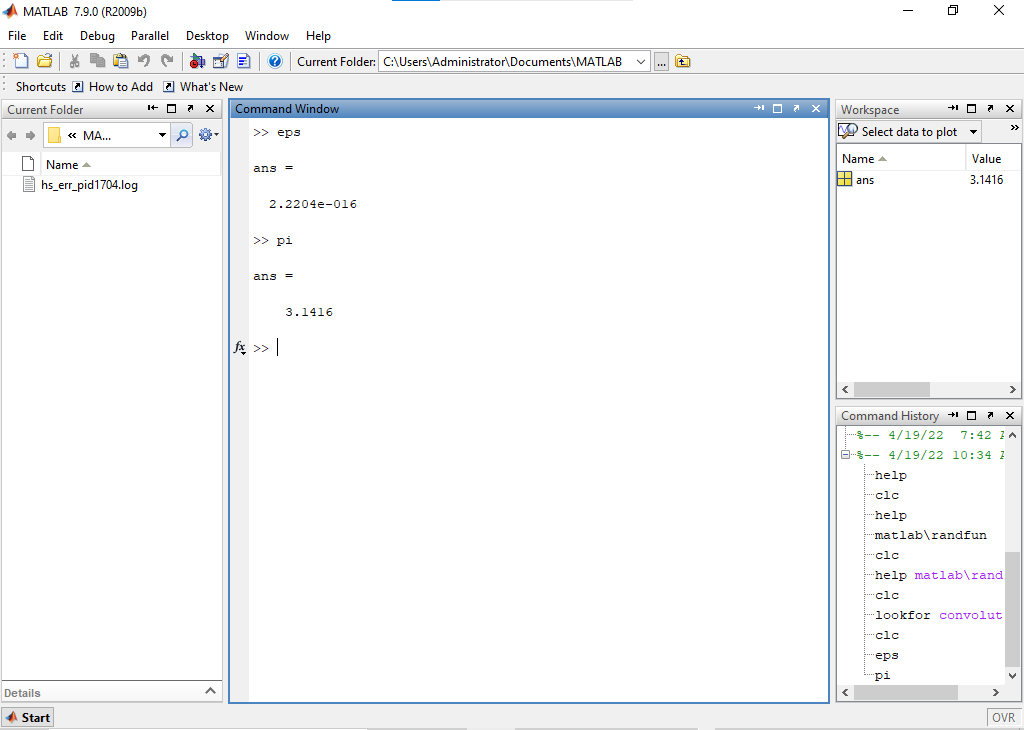
eps =

2.2204e‐16

>> pi

ans =

3.1416



**1.5.1 Variable Assignment:**

The equality sign is used to assign values to variables:

>> x = 3

x =

3

>> y = x^2

y =

9

Variables in MATLAB are case sensitive. Hence, the variables "x" and "y" are distinct from "X"

and "Y" (at this point, the latter are in fact, undefined).

Output can be suppressed by appending a semicolon to the command lines.

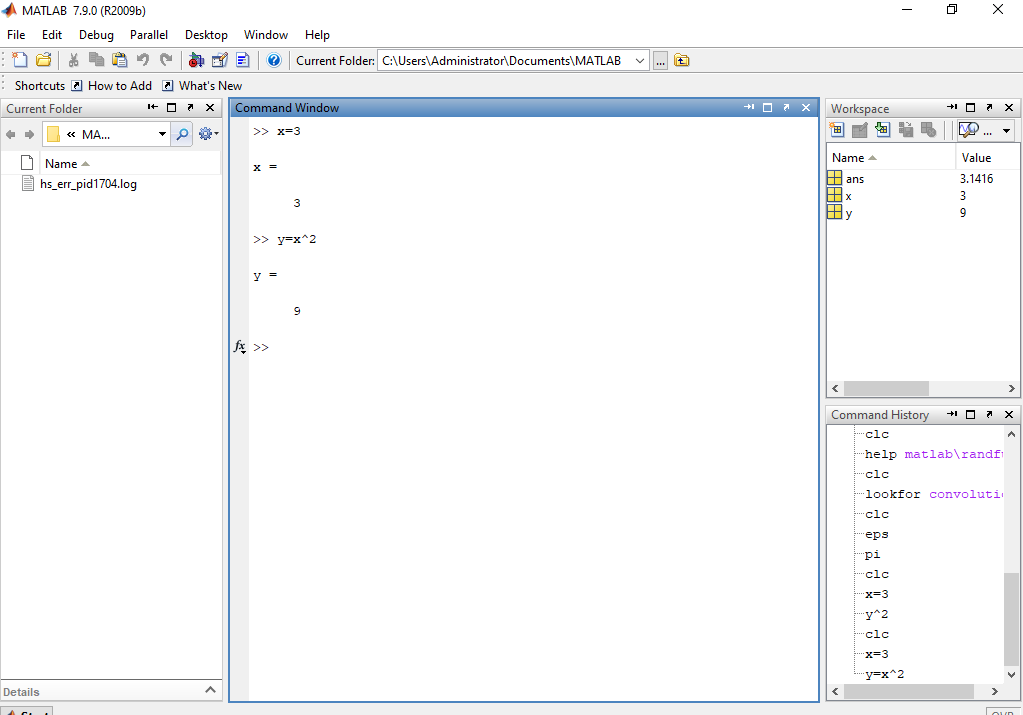
>> x = 3;

>> y = x^2;

>> y

y =

9



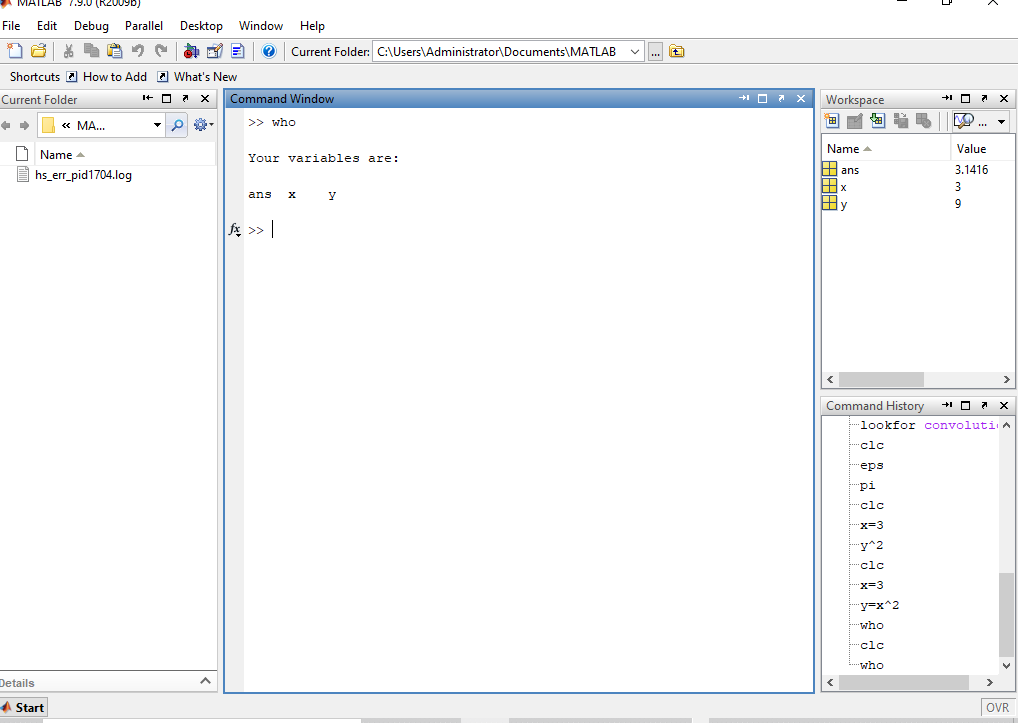
**1.5.2 Active Variables:**

At any time, you want to know the active variables you can use who:

>> who

Your variables

are: ans x y



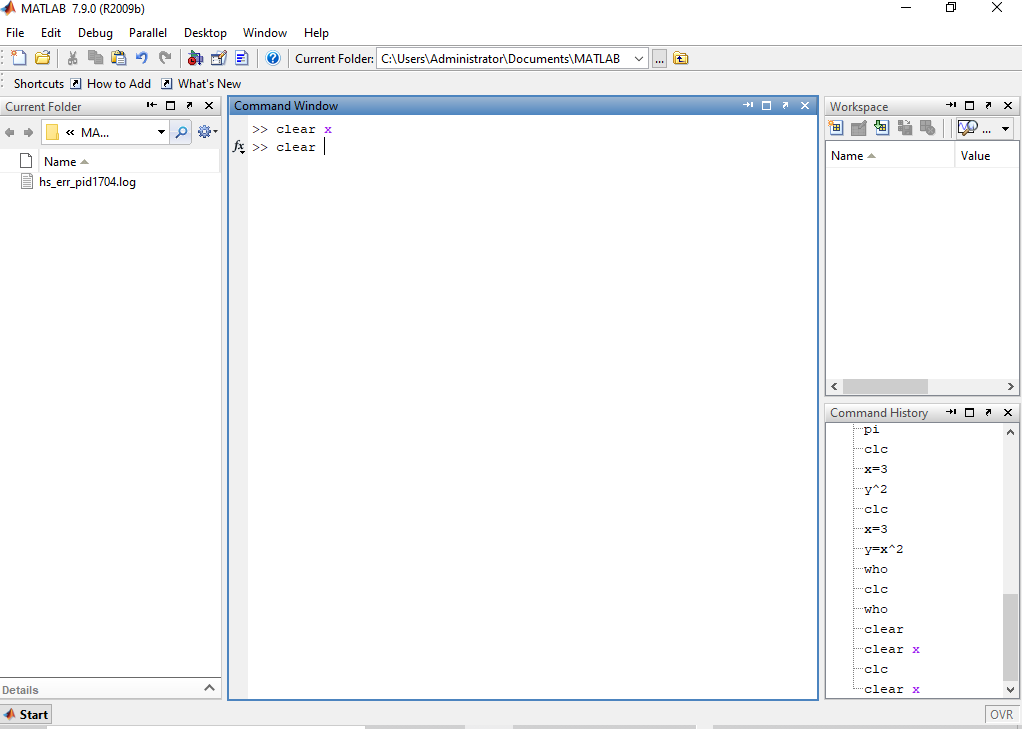
**1.5.3 Removing a Variable:**

To remove a variable, try

this: >> clear x

To remove all the variables from workspace, use

clear >> clear



**1.5.4 Saving and Restoring Variables:**

To save the value of the variable "x" to a plain text file named "x.value"

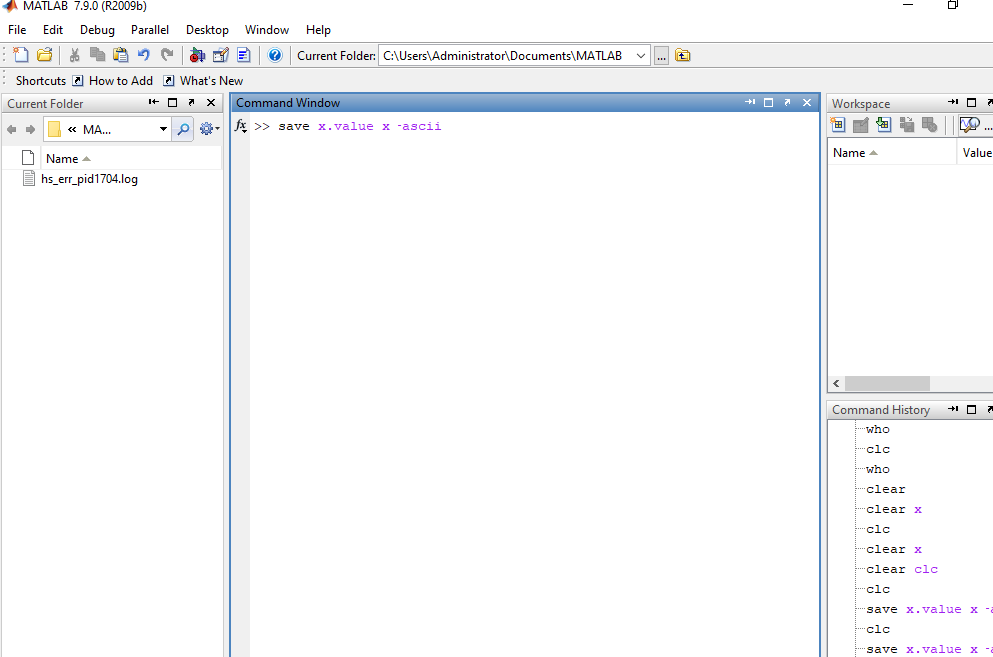
use >> save x.value x ‐ascii

To save all variables in a file named mysession.mat, in reloadable format, use

>> save mysession

To restore the session, use

>> load mysession



**1.6 VARIABLE ARITHMETIC:**

MATLAB uses some fairly standard notation. More than one command may be entered on a single line,

if they are separated by commas.

>> 2+3;

>> 3\*4, 4^2;

Powers are performed before division and multiplication, which are done before subtraction and

addition. For example

>> 2+3\*4^2;

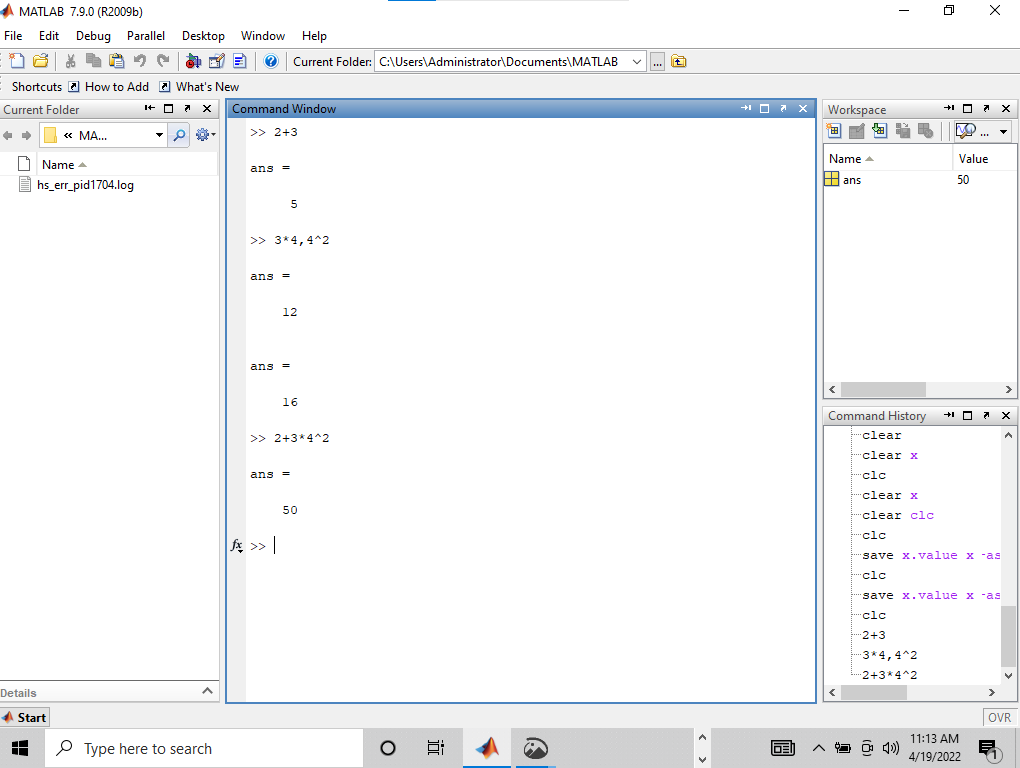
generates ans = 50. That is:

2+3\*4^2 ==> 2 + 3\*4^2 <== exponent has the highest

precedence ==> 2 + 3\*16 <== then multiplication operator

==> 2 + 48 <== then addition operator

==> 50



**1.6.1 Double Precision Arithmetic:**

All arithmetic is done to double precision, which for 32‐bit machines means to about 16 decimal

digits of accuracy. Normally the results will be displayed in a shorter form.

>> a = sqrt(2)

a =

1.4142

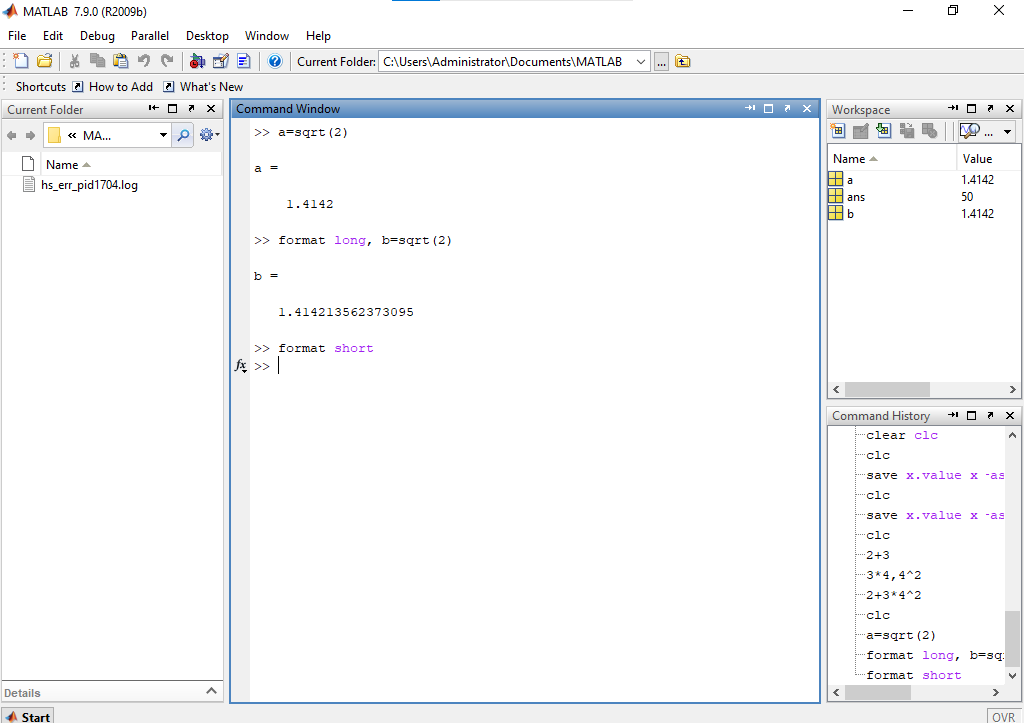
>> format long, b=sqrt(2)

b =

7

1.41421356237310

>> format short



**1.6.2 Command-Line Editing:**

The arrow keys allow "command‐line editing," which cuts down on the amount of typing

required, and allows easy error correction. Press the "up" arrow, and add "/2." What will this

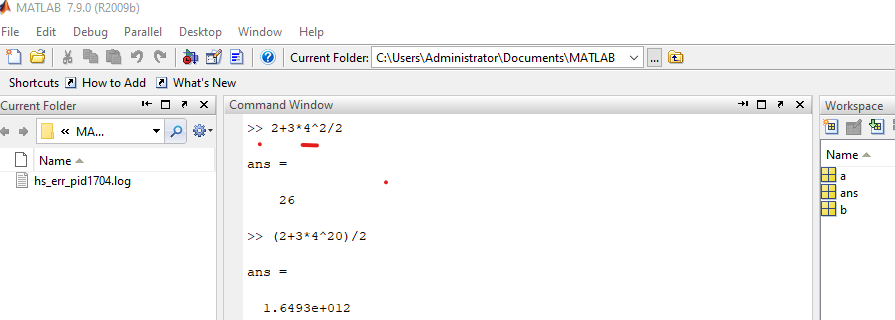
produce?

>> 2+3\*4^2/2

Parentheses may be used to group terms, or to make them more readable. For

example: >> (2 + 3\*4^2)/2

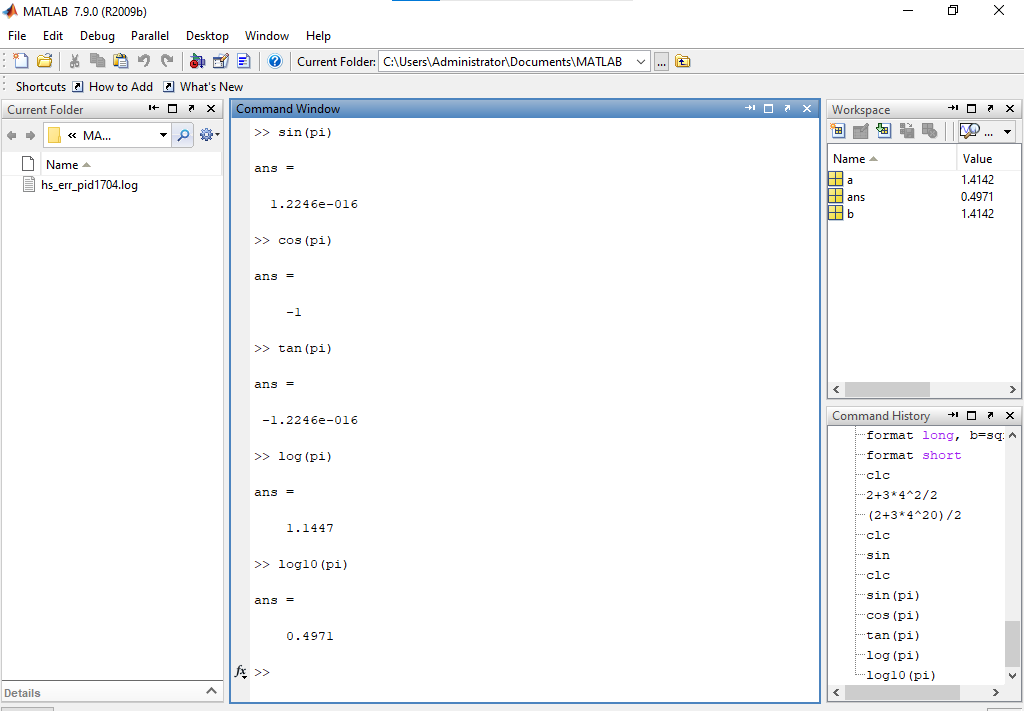
generates ans = 25.



**1.6.3 Built-In Mathematical Functions:**

MATLAB has a platter of built‐in functions for mathematical and scientific computations. Here is a

summary of relevant functions.



**1.7 TIMING COMMANDS:**

Timing functions may be required to determine the time taken by a command to execute or an

operation to complete. Several commands are available to accomplish it:

**1.7.1 Clock:**

CLOCK returns Current date and time as date vector. CLOCK returns a six-element date vector

containing the current time and date in decimal form:

CLOCK = [year month day hour minute seconds]

The first five elements are integers. The second’s element is accurate to several digits beyond the

decimal point. FIX(CLOCK) rounds to integer display format.

**1.7.2 Etime:**

ETIME Elapsed time.

ETIME(T1,T0) returns the time in seconds that has elapsed between vectors T1 and T0. The two

vectors must be six elements long, in the format returned by CLOCK:

T = [Year Month Day Hour Minute Second]

Time differences over many orders of magnitude are computed accurately. The result can be

thousands of seconds if T1 and T0 differ in their first five components or small fractions of

seconds if the first five components are equal.

t0 = clock;

operation

etime(clock,t0)

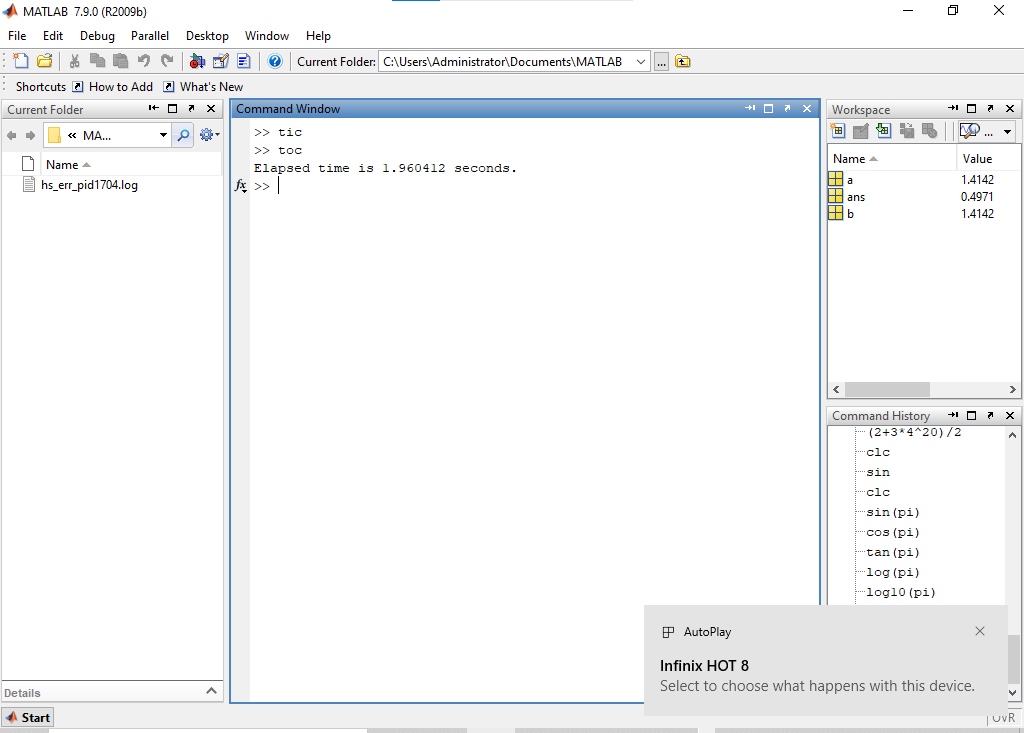
**1.7.3 Tic Toc:**

TIC Start a stopwatch timer.

The sequence of commands

TIC, operation, TOC

prints the number of seconds required for the operation.



**1.8 INPUT & DISPLAY:**

**1.8.1 INPUT:**

INPUT prompts for user input.

R = INPUT ('How many apples')

gives the user the prompt in the text string and then waits for input from the keyboard. The input

can be any MATLAB expression, which is evaluated, using the variables in the current workspace,

and the result returned in R. If the user presses the return key without entering anything, INPUT

returns an empty matrix.

Example: Entering a single variable

>> x=input('Enter a variable: ')

Enter a variable: 4

x =

4

Example: Entering a vector

A vector is entered by specifying [] and elements are inserted inside these brackets, separated by

space.

>> x=input ('Enter a vector: ')

Enter a vector: [3 4 1]

x =

3 4 1

Example: A \n entered after the string results in starting a new line.

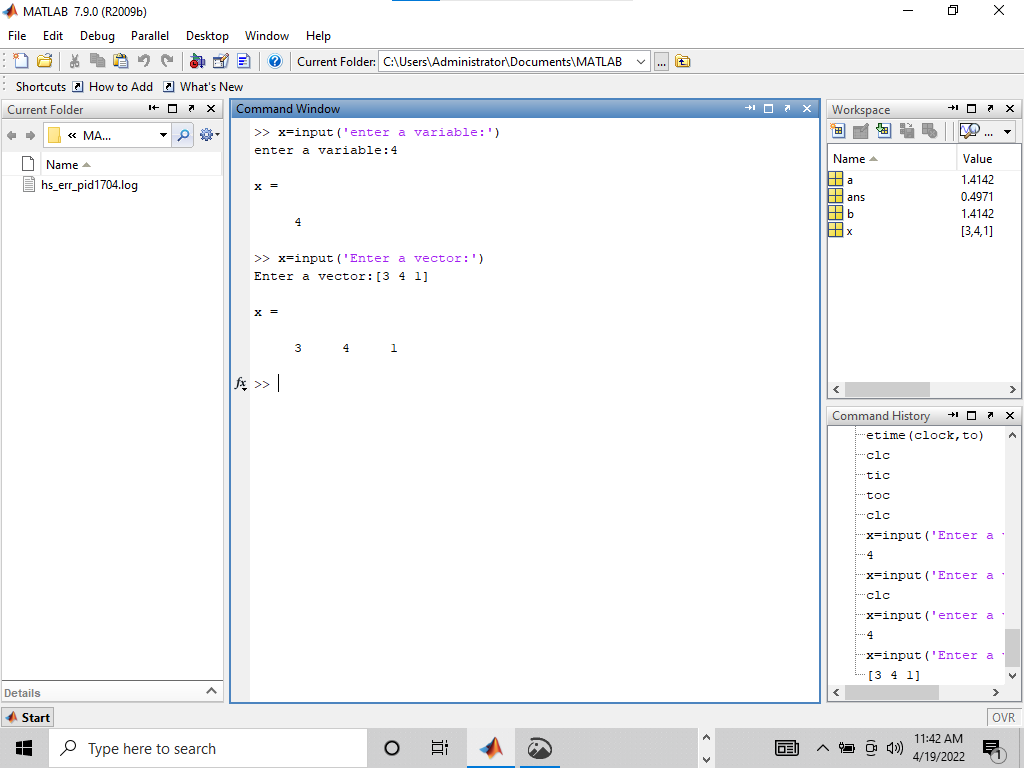
>> x=input ('Enter a value\n')

Enter a value

5

x =

5



**1.8.2 DISP:**

DISP(X) displays the value of variable X, without printing the variable name. In all other ways it's

the same as leaving the semicolon off an expression.

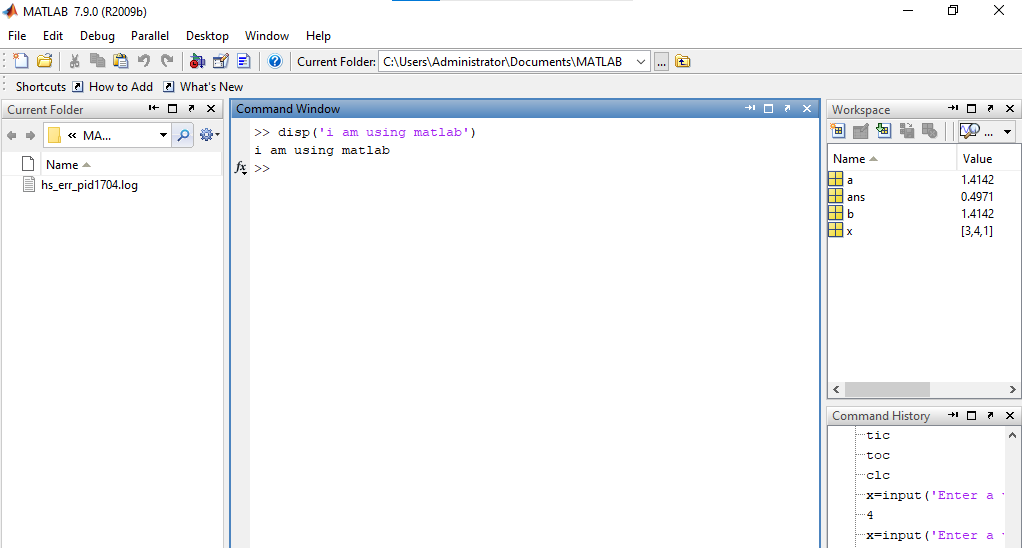
DISP(‘string’) is another variation of the same function that is used to display a string on

the command prompt.

Example:

>> disp('I am using MATLAB')

I am using MATLAB



**1.9 M-Files:**

Typing errors are time‐consuming to fix if you are working in the command window because you need

to retype all or part of the program. Even if you do not make any mistakes, all of your work may be lost

if you inadvertently quit MATLAB. To preserve large sets of commands, you can store them in a special

type of file called an M‐file. MATLAB supports two types of M‐files: script and function M‐files. To hold a

large collection of commands, we use a script M‐file. The function M‐file is discussed in coming lab. The

script file has a '.m' extension and is referred to as an M‐file (for example, myfile.m

myfuncion.m, etc.). The commands in the script file can then be executed by typing the file name

without its extension in the command window. Commands in a script utilize and modify the contents of

the current workspace. It is possible to embed comments in a script file.

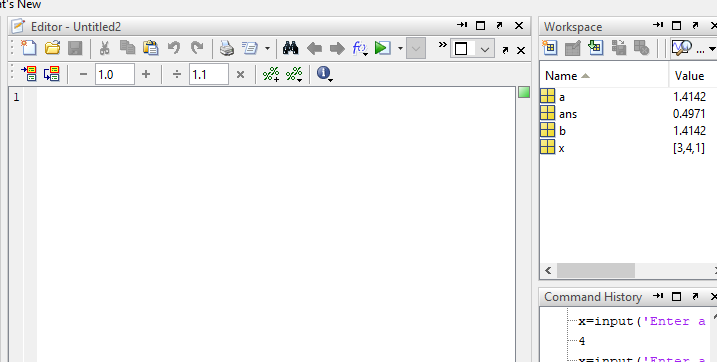
To make a script M‐file, you need to open a file using the built‐in MATLAB editor. There are two ways

to accomplish it:

1. From file menu, click NEW

2. Type edit on command line

A new window appears like one shown in the figure below.



**-------------------------TASK 01--------------------------**

a. Matlab stores numeric data as double‐precision floating point by default. To store data

as an 8‐ bit integer, int8 (a conversion function) can be used. Type the sample code in

matlab command window:

>> x = 26

>> whos

>> y = int8(x)

>> whos

What difference do you see? State your findings. (Also try uint16, uint32, uint64)

**Problem Analysis:**

This problem requires explicitly type casting data from float (in part a) to intX and from string (in part b) to int8. We will be using built-in functions, intX where X is one of (8, 16, 32, 64) and indicates the bits that the target integer will have. By default, all numeric values in MATLAB are stored as 8-bit doubles. Another built-in function used is whos() which provides the name, size, Bytes, type (class) and Attributes of all the variables in the program. We will be particularly interested in the Bytes and class parameters for this problem.

Algorithm:

Step 1) Assign 26 to x.

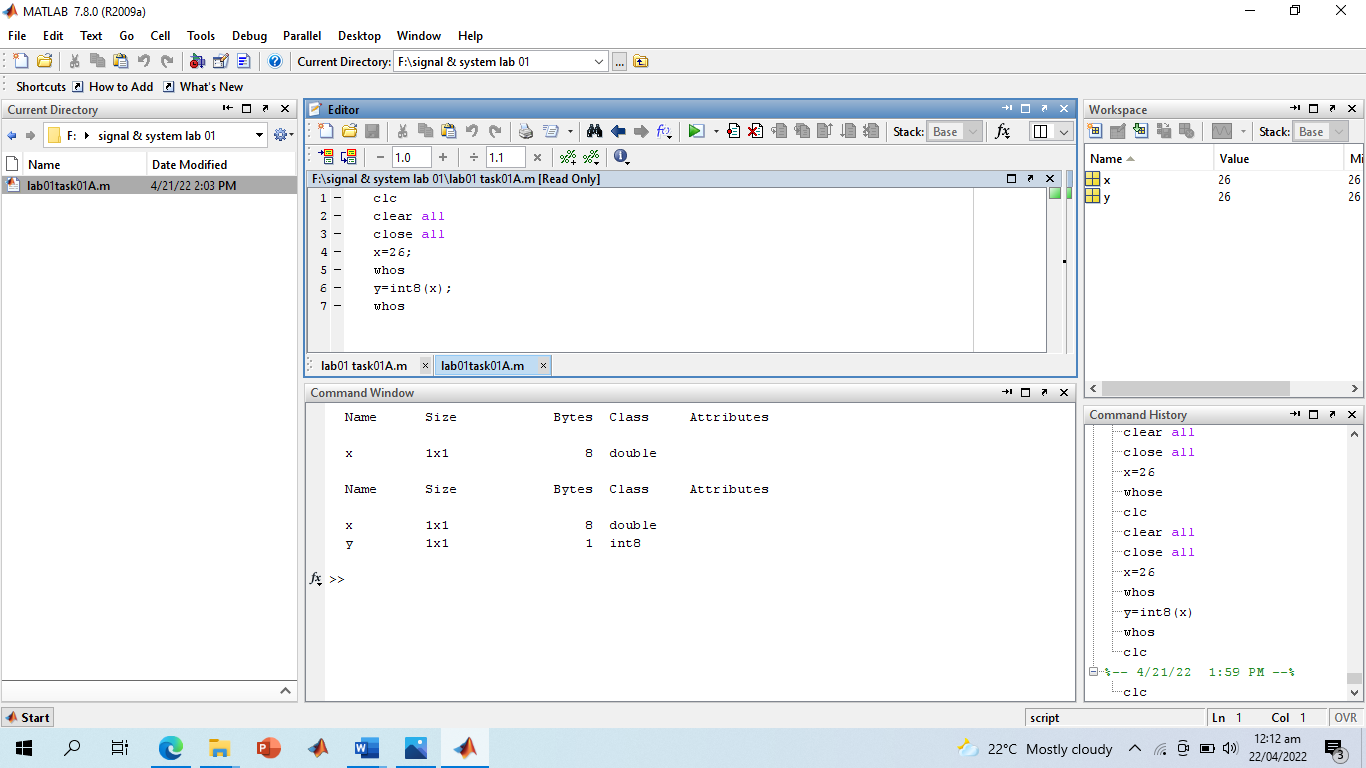
Step 2) Call int8 functions and pass x as argument and then assign returned value to y.

Step 3) Call whos() function

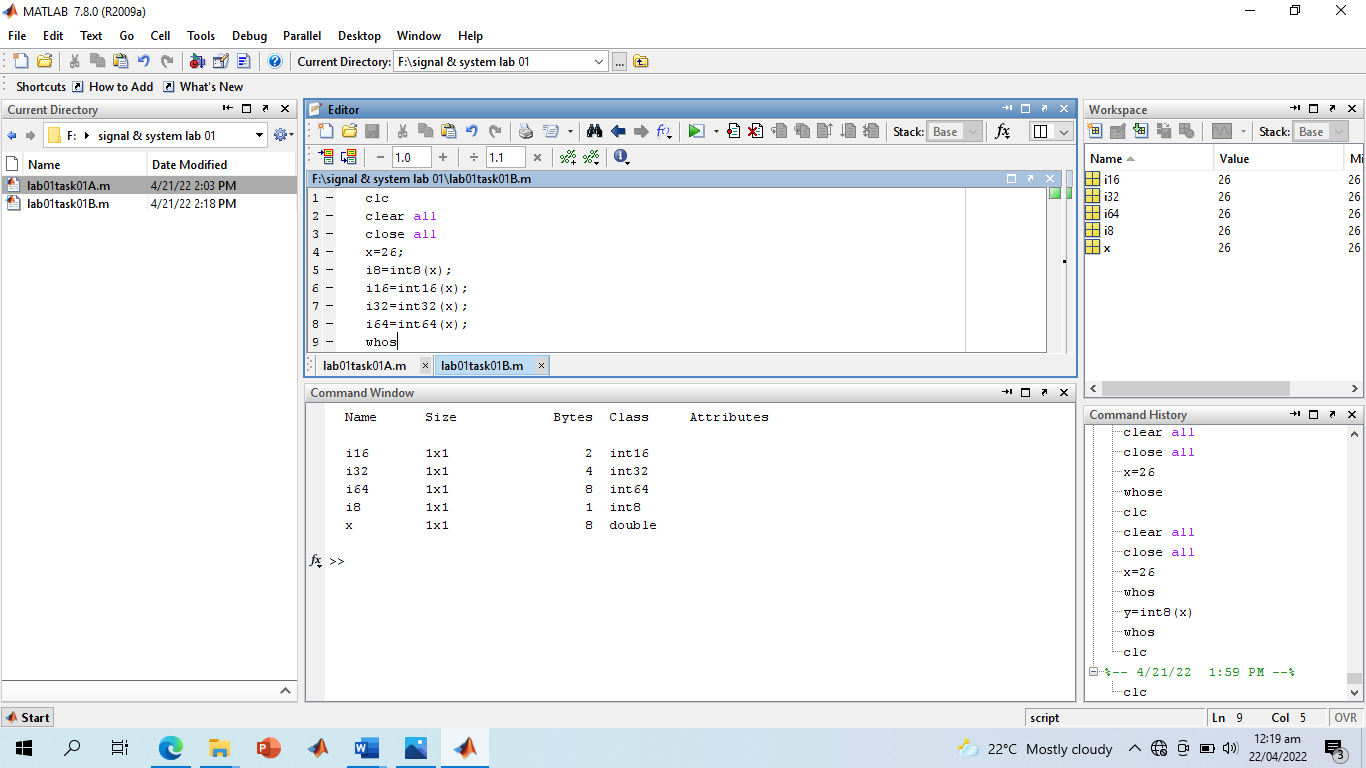
Step 4) Repeat Step2 for int16, int32, int 64 and then execute Step 3.

Step 5) Repeat from Step1 to 3 but assign „ABDUL WAHAB‟ to x.

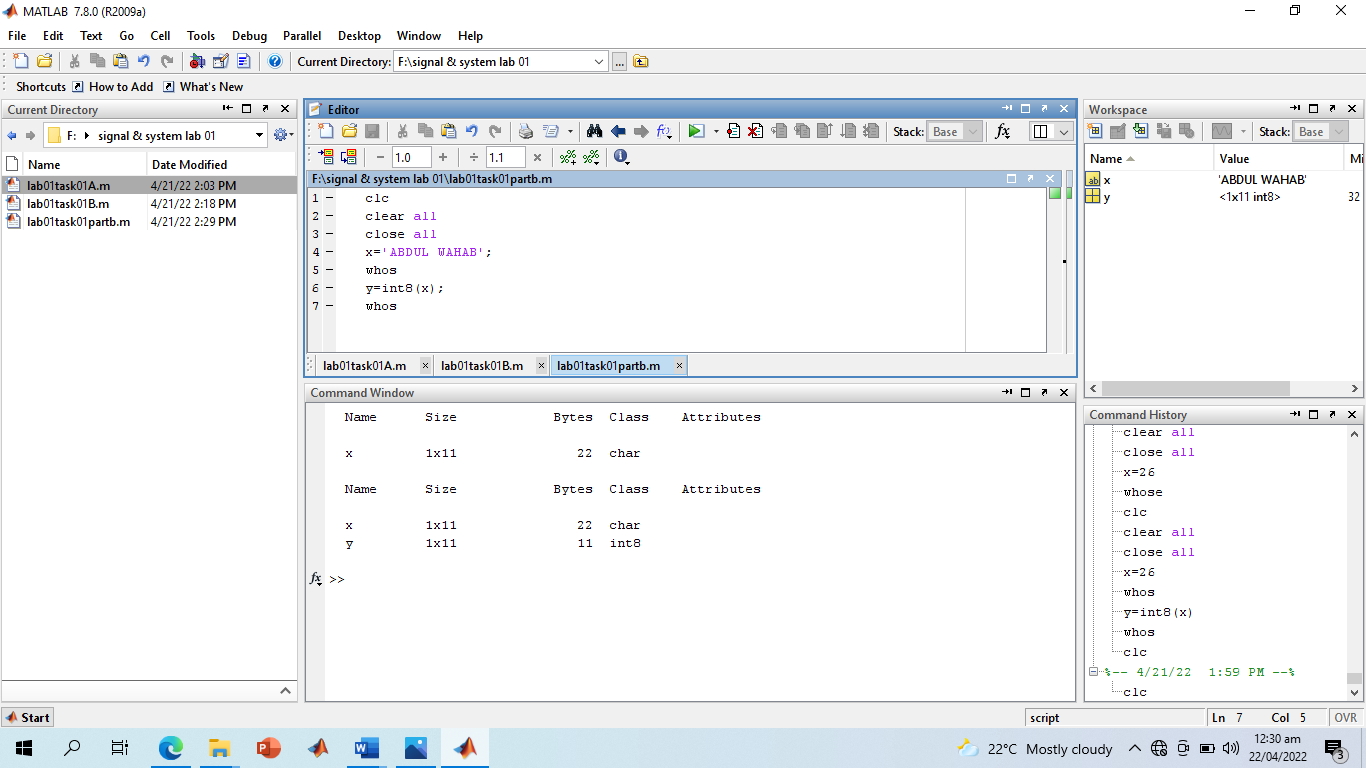
**Screenshot of Source Code and Output:**



**Lab o1 Task 01 Part a (portion i)**



**Lab o1 Task 01 Part a (portion ii)**



**Lab o1 Task 01 Part b**

**Discussion and Conclusion:**

We saw that the type casting in MATLAB is not same as in other languages. Since every variable in MATLAB is an array, we see that when we converted string into int8, then, because the string (ABDDUL WAHAB) had 11 characters, the resulting integer value was a 1x11 matrix.

**-------------------------TASK 02--------------------------**

Create an M-File to prove trigonometric expressions and find the time taken by each. Problem Analysis: This problem requires use of simple trigonometric formulas and the time taken by their execution. For keeping track of time, I will be using tic toc function. The tic returns current time while toc(parameter) returns time elapsed since parameter in seconds. I have written a script that calculates all the ten expressions. I first note the current time in t0 by assigning it to tic. Then I store the result of LHS in direct variable and then, to find the time taken by this direct expression, pass t0 to toc and assign the returned value to time\_Taken\_For\_First. Then I reassign tic to t0, store the result of RHS in indirect and then pass tic to toc and assign the return value to time\_Taken\_For\_Second which now stores time taken by indirect. Then I use fprintf () for printing since its more convenient and easier to format.

**Algorithm:**

Step 1) Assign tic to t0

Step 2) Calculate the LHS and store result in direct.

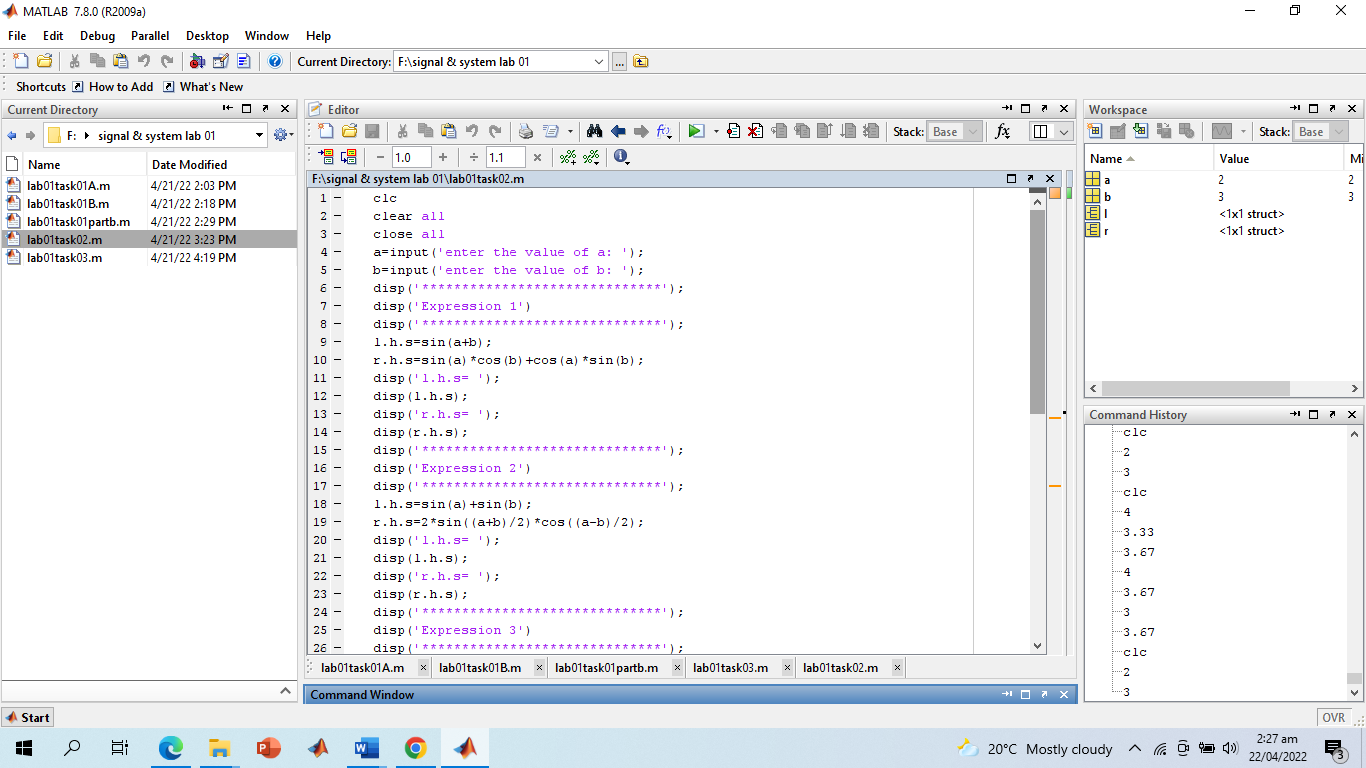
Step 3) Pass t0 to toc and assign the returned value to time\_Taken\_For\_First

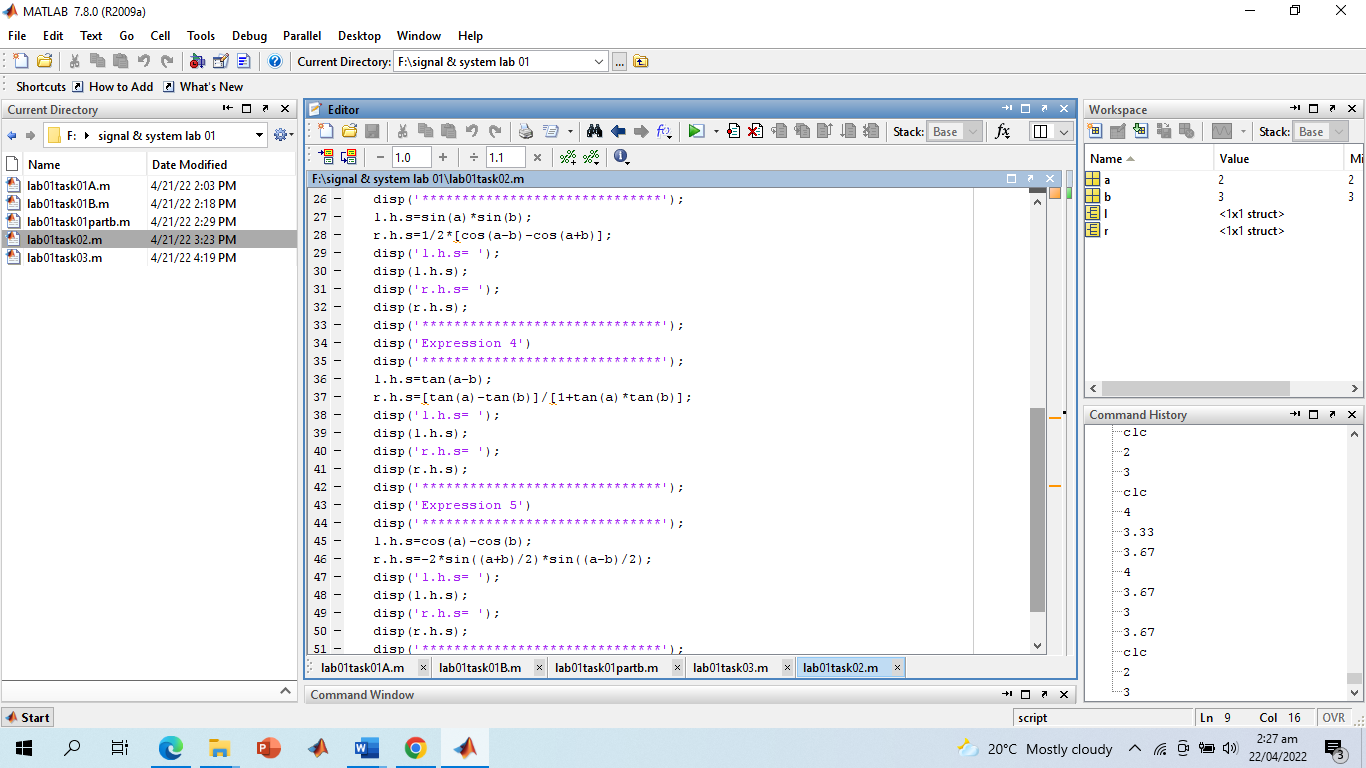
. Step 4) Reassign tic to t0.

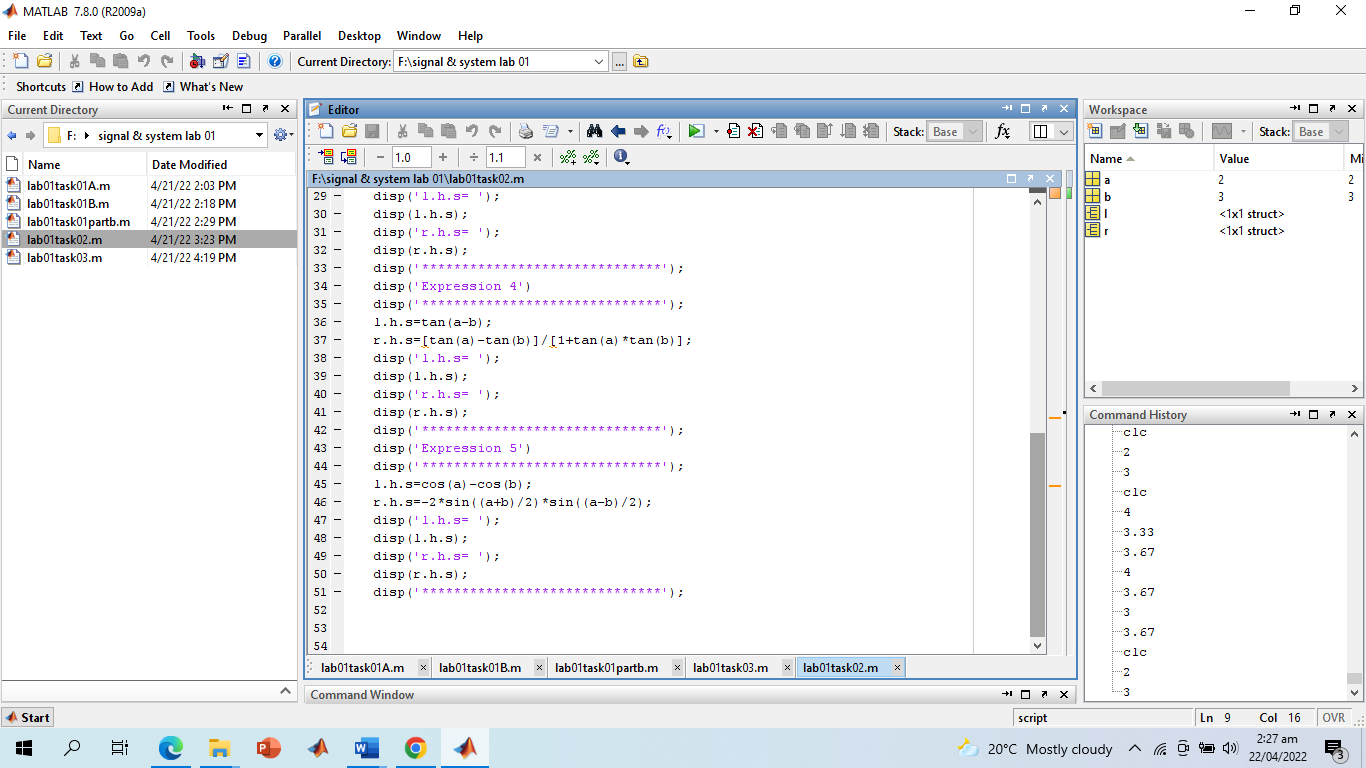
Step 5) Calculate RHS and store result in indirect. Step 6) Pass t0 to toc and assign the returned value to time\_Taken\_For\_Second

. Step 7) Display LHS, RHS, time\_Taken\_For\_First and time\_Taken\_For\_Second. Step 8) Repeat 1 to 7 10 times replacing RHS and LHS with corresponding equations.

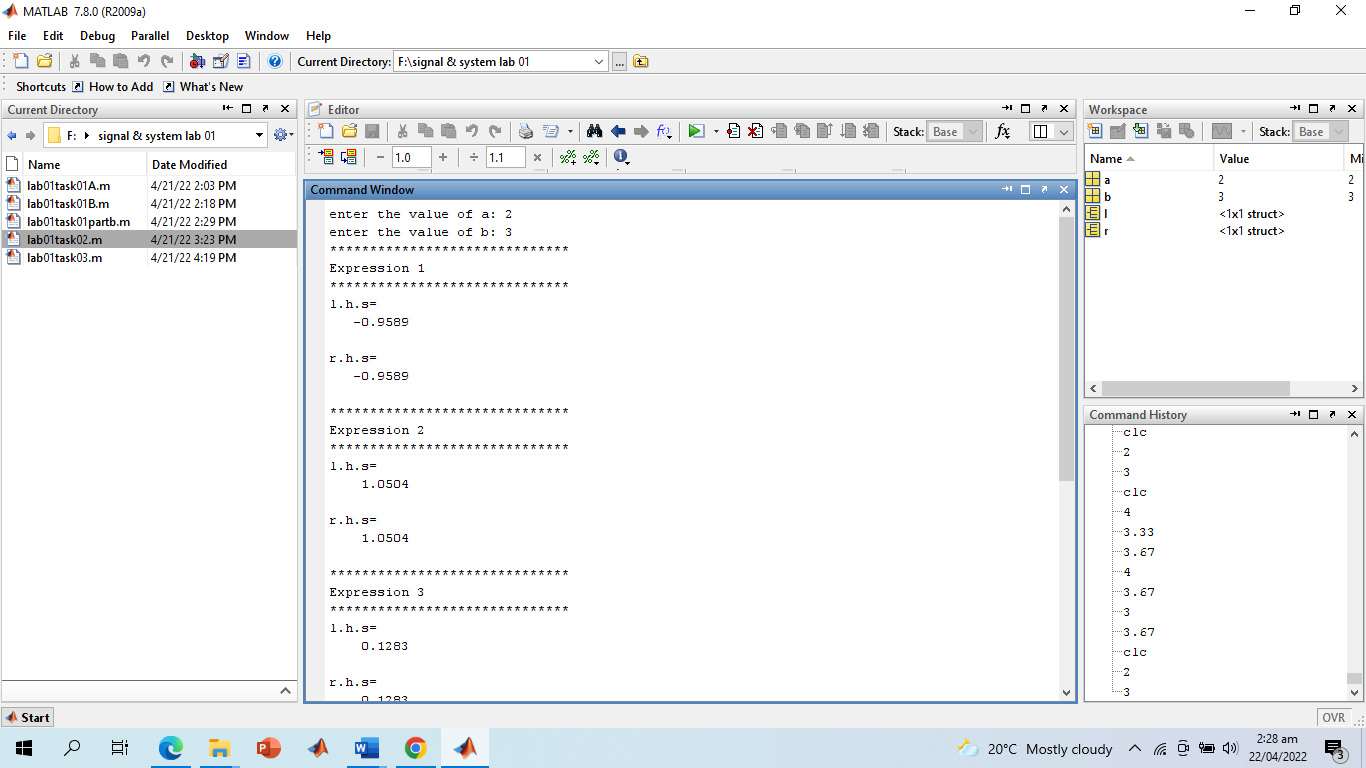
**Screenshot of Source Code:**

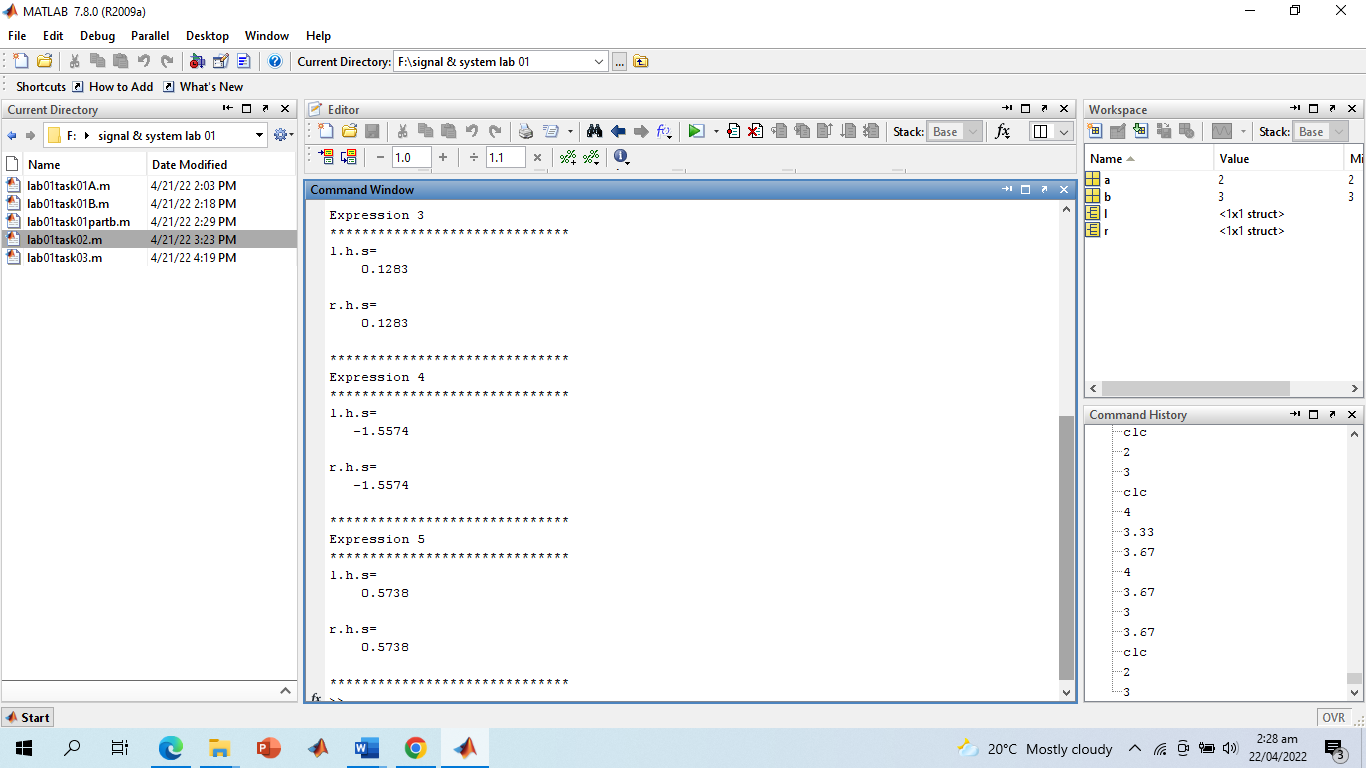






**Screenshot of Output:**





**Discussion and Conclusion:**

We note that in some cases the LHS takes less time while in others, the RHS. One other thing that may be noted is that I used %g modifier instead of the common %f. This is because MATLAB usually shows floats to 6 decimal places even if it’s a round figure. For instance, say a float “a” equals 3, then %f would show 3.000000. To avoid this, I used %g which automatically truncates insignificant zeros.

**------------------------- TASK 03 --------------------------**

Write a CGPA Calculator program using M-File: Design a transcript for your second semester result i.e., take grade points and credit hours of each subject as input from user and store in variables. Take product of each subject grade points with its credit hours and divide by total credit hours in order to evaluate CGPA. Show the results in the form of well-designed transcript. Use the following table to display equivalent grades for each 13-grade point

**Problem Analysis:**

I will be developing a GPA calculator for 3rd semester which had 5 courses where 3 had labs too so in total our 3rd semester had 8 subjects. This problem can be solved using arrays and loop although it can be solved without these constructs, still using loops and arrays will help generalize the problem. Firstly, we need to take the grades from user. For this, instead of writing many input statements, I wrote a for loop that runs 8 times (remember we had 8 courses in 3rd semester). As we go on to write for loop, the first problem we encounter is of the message that displays while we take input from user. We want to show the following message to user for each subject, enter your grade for [course title] where course title should change in each iteration. So, the obvious answer is to store the titles in an array of strings and then have our loop take those strings. For some reasons, I found using cells more convenient than arrays. So now we have a cell CourseTitles that stores names of all subjects. What we really want from our loop is to calculate SGPA that is the sum of products of subject’s grade and its credit hours. For this, we also need to store credit hours of each subject. So, we create an array creditHours that stores credit hours of each subject. Note that the sequence is same as that in CourseTitles cell that is if first element in CourseTitles is OOP Theory and the first element in creditHours is 3, then OOP Theory had 3 credit hours. Now we have a way of showing the names of each subject in loop and we also know how many credit hours does the subject of current iteration has. The next is receiving the input from user. Whilst we can get input using the built-in input function, we definitely need to check if the value entered by the user is a valid grade or not since grades have to be certain fixed numbers like 4, 3.67, 3.33 and not random numbers like 2.56 or so. To overcome this, we create an array gradesInDigits that stores numerical values of all possible grades a student may obtain in a subject. Now we know how to get input, filter that input and then multiply the grade by credit hours. However, in final transcript message, I want to display the grades in letters not numbers that is if some student received a 3.33 in DLD then I want to display B+ in front of DLD rather than the number. Now we need a mapping from numerical grades to letters. We already have an array for numerical grades. If we create an array for grades using letters, that stores grades in the same sequence as the numerical array than we can easily map between the two. This is precisely what I have done.

**Algorithm:**

Step 1) Create an array and a cell for storing course titles and their credit hours. The two should have the same sequence.

Step 2) Also create an array for storing grades of individual subjects. Initialize it with „F‟.

Step 3) Create an array and a cell for storing possible achievable grades in digits and in letters. The two should have the same sequence.

Step 4) Initialize SGPA to 0.

Step 5) Run a loop 8 times where the counter variable will begin from 1 and will get incremented by 1 after each iteration.

Step 6) Concatenate the string “Enter your grade for” with the element in course title cell corresponding to the current iteration.

Step 7) Next, receive numeric input from user and check if the value exists in the grades in digits array. Jump to step 10 if the user input does not exist in array.

Step 8) If step 7 returns true, then multiply the grade by the credit hours of the corresponding subject and add this product to SGPA and run

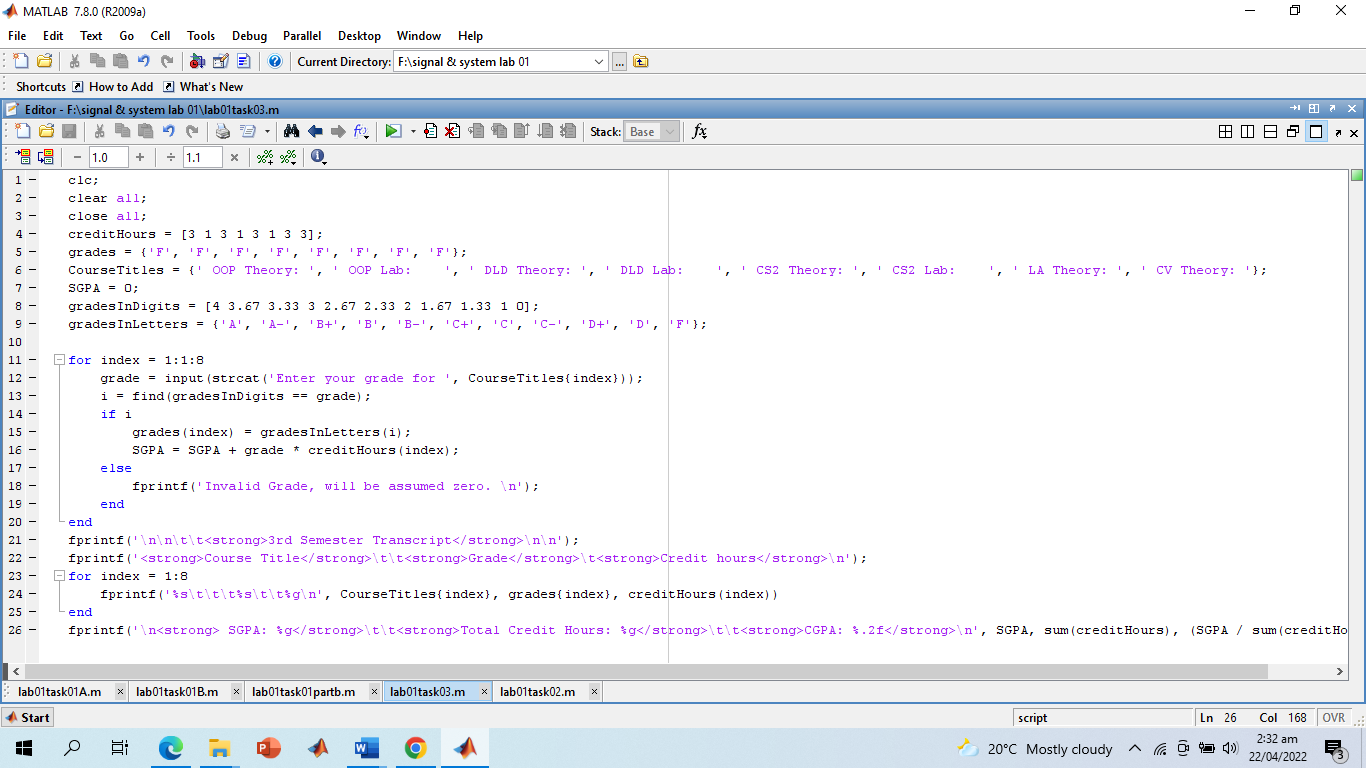
step 9. Step 9) Find the corresponding letter of the grade and store in the grades cell and jump to step 11.

Step 10) If the step 7 returns false. Then consider the value invalid and assume grade to be zero. We already have „F‟ stored in grades cell so just display an invalid input message.

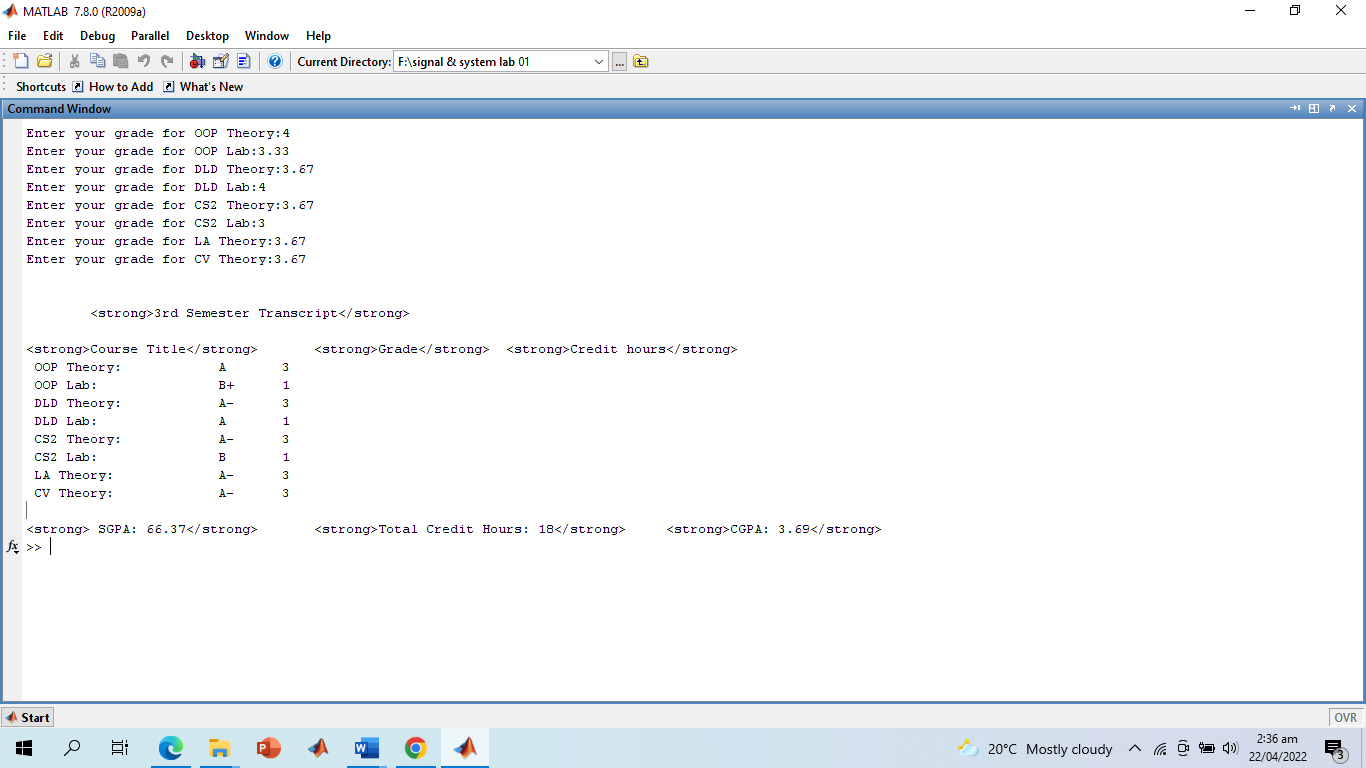
Step 11) Iteration completed. Now increment the counter variable and perform next iteration.

Step 12) After loop’s execution, display the grades stored in grades cell against their corresponding subjects‟ name. Step 13) Calculate CGPA by dividing SGPA by sum of credit hours array.

**Screenshot of Source Code:**



**Screenshot of Output:**



**Discussion and Conclusion**:

There were a few built-in functions that we used. The find function returns and array of indices of all the elements whose value match a specific value. The sum function returns the sum of all the elements of an array. I used the markup tag to bold out certain text. The code is quite generalized in that you can add or replace or remove subjects in arrays and the rest of the code would work properly. One more thing that can be done to create it more generalized is that instead of hard coding the upper limit of counter variable in loop, 8 in this case, we can have it set to the length of credit hours array.

**-------------------------- TASK 04 --------------------------**

Write a simple code to swap the values of two variables of double type using M-file. Create the logic in such a way that no third variable is used. Show the etime for this code.

**Problem Analysis:**

This problem is fairly straight forward. We need to note time at the beginning of our program. For this we initialize t0 to clock. Next, we take to inputs from users and swap them without third variable. Then we find the time elapsed since t0 zero by passing t0 as the second argument to etime whose first argument is clock. As for swapping without third variable, we need to store the sum of two variables being swapped in one of them. Then store the result of difference of this sum and the other variable. Next, we need to subtract the other variable from the sum.

**Algorithm:**

Step 1) Initialize t0 to clock.

Step 2) Receive two inputs from user var1 and var2.

Step 3) Store the sum of var1 and var2 in var1.

Step 4) Now store the difference of var1 and var2 in var2

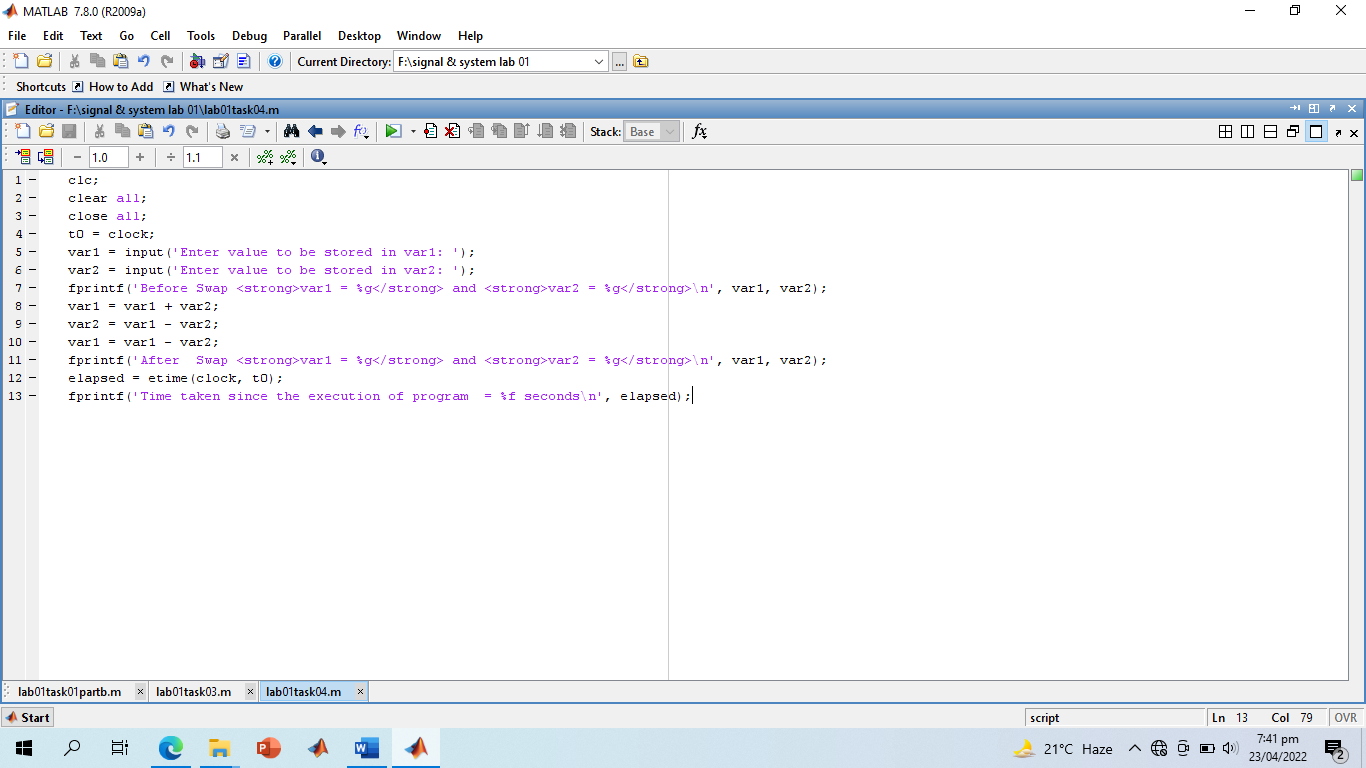
. Step 5) Now again store the difference of var1 and var2 in var1.

Step 6) Display the swapped values.

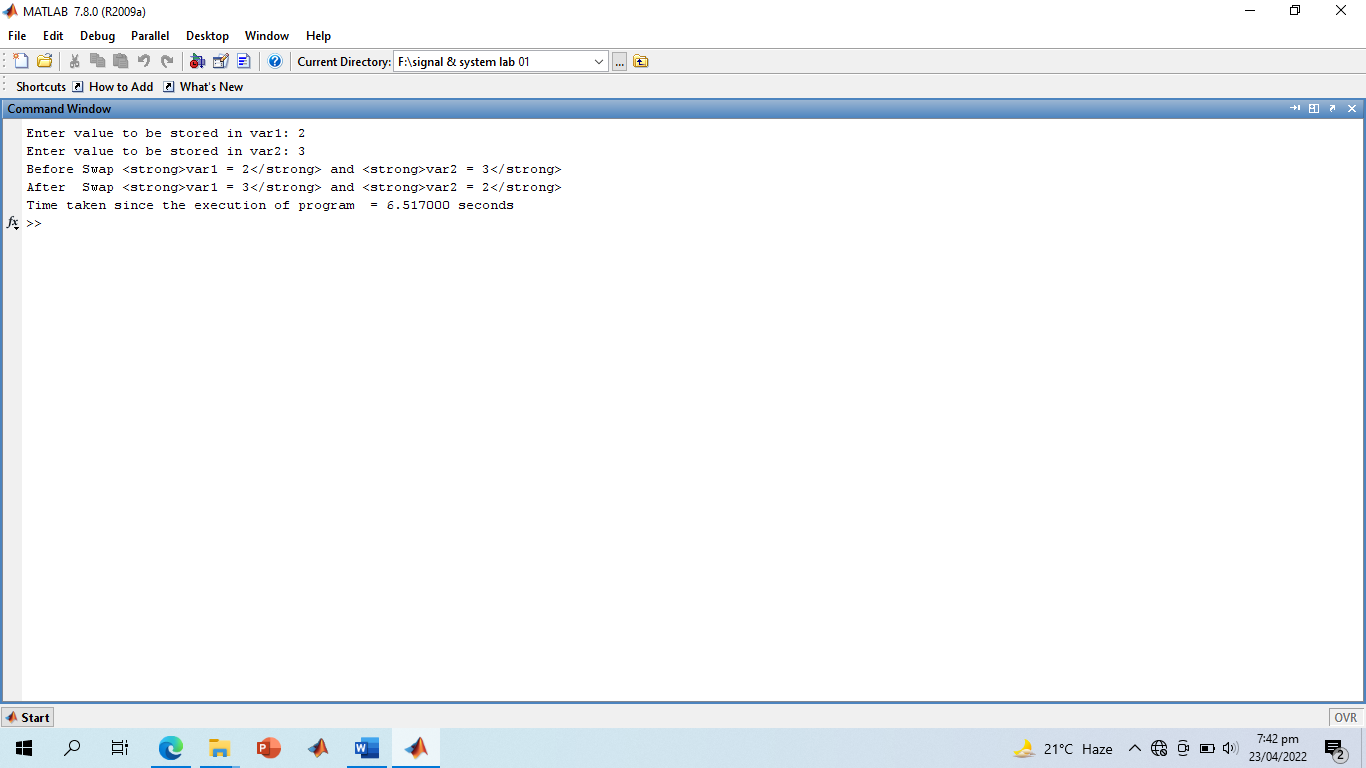
Step 7) Pass clock and t0 as arguments to etime and store return value in elapsed.

Step 8) Display elapsed.

**Screenshot of Source Code:**



**Screenshot of Output:**



**-------------------------- TASK 05 --------------------------**

Implement the Pythagoras theorem in MATLAB that takes input from the user.

**Problem Analysis:**

This is a fairly straight forward problem. We need to take the values of base and perpendicular from the user and then calculate hypotenuse.

Since, ℎ𝑦𝑝𝑜𝑡𝑒𝑛𝑢𝑠𝑒 = 𝑏𝑎𝑠𝑒 2 + 𝑝𝑒𝑟𝑝𝑒𝑛𝑑𝑖𝑐𝑢𝑙𝑎𝑟

This means we need to take the square and of two values and then square of their sum. The ^ operator is used in MATLAB to raise a base to some exponent. We can use this to find the square. Also, the sqrt built-in function returns the square root of the argument.

**Algorithm:**

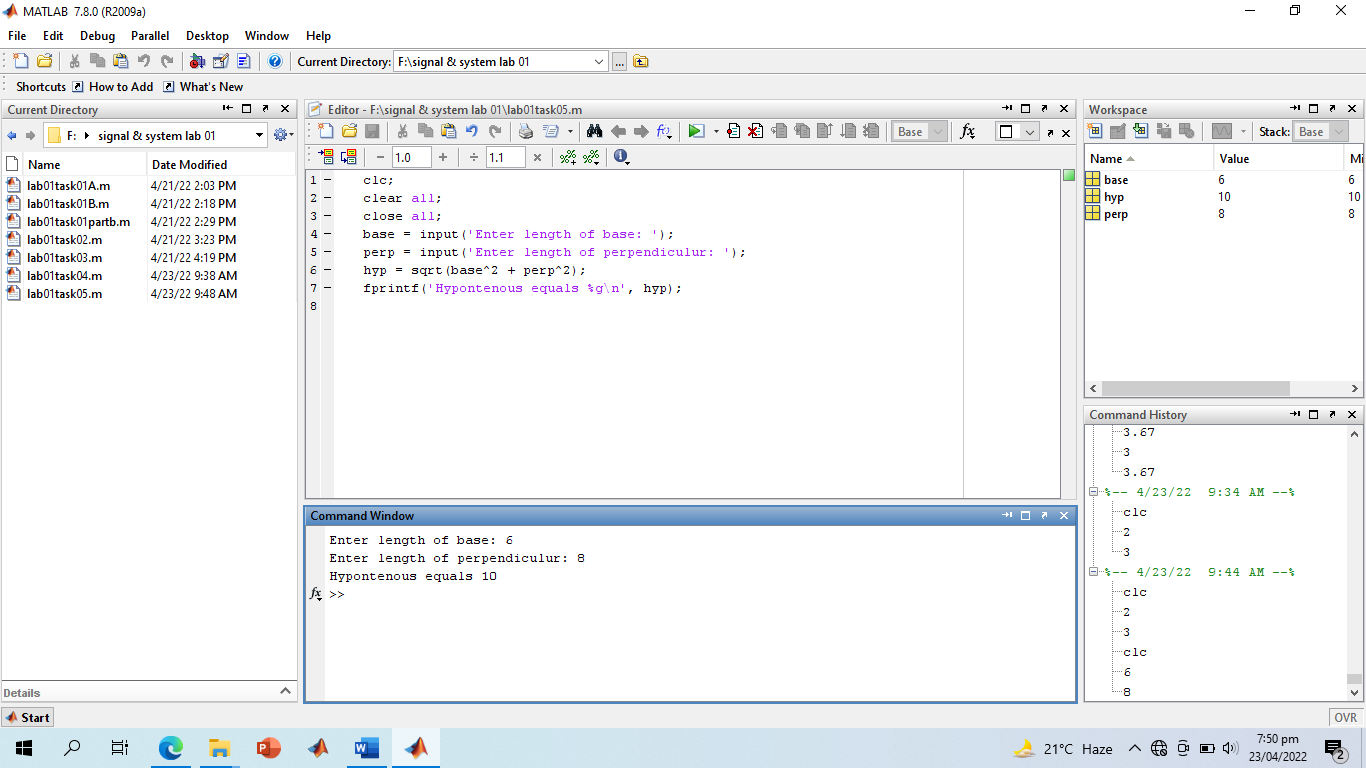
Step 1) Take values of base and perpendicular from user.

Step 2) Using ^ operator, find the squares of base and perpendicular.

Step 3) Use the sqrt function to find the square root of sum of the squares of base and perpendicular.

Step 4) Store the value returned by sqrt in hyp and display it.

**Screenshot of Source Code and Output:**



**Discussion and Conclusion:**

Again, our task is done but there is something more we can do. Since MATLAB provides very rich graphical tools for drawing, I think it would be nice to draw an actual triangle of the lengths equal to the lengths we just calculated.

**-------------------------- TASK 06 --------------------------**

Implement a temperature conversion scenario in Matlab that takes the temperature from the user in Fahrenheit and displays the output in Centigrade.

**Problem Analysis:**

We just need to take temperature in Fahrenheit and convert it into centigrade using the following formula. 𝐶 = 𝐹 − 32 × 5 9 On conversion side, there is no issue. However, I want to display it in the following format, x° Fahrenheit is equal to y° centigrade. For this I need to display the degree sign. Since we have covered basic data types in MATLAB, I think it would be nice to make use of them. My keyboard does not have a degree sign and if we try to copy the degree sign into fprintf statement, if will fail to parse it as recognizable character. So, the only way of doing it is assigning character code. Since we know all characters have a Uni Code, we can use the uni code of degree sign (176) to view it. The char () function in MATLAB, if provided with a numerical value, returns the character corresponding to that ASCII code. So, char (176) will return degree sign.

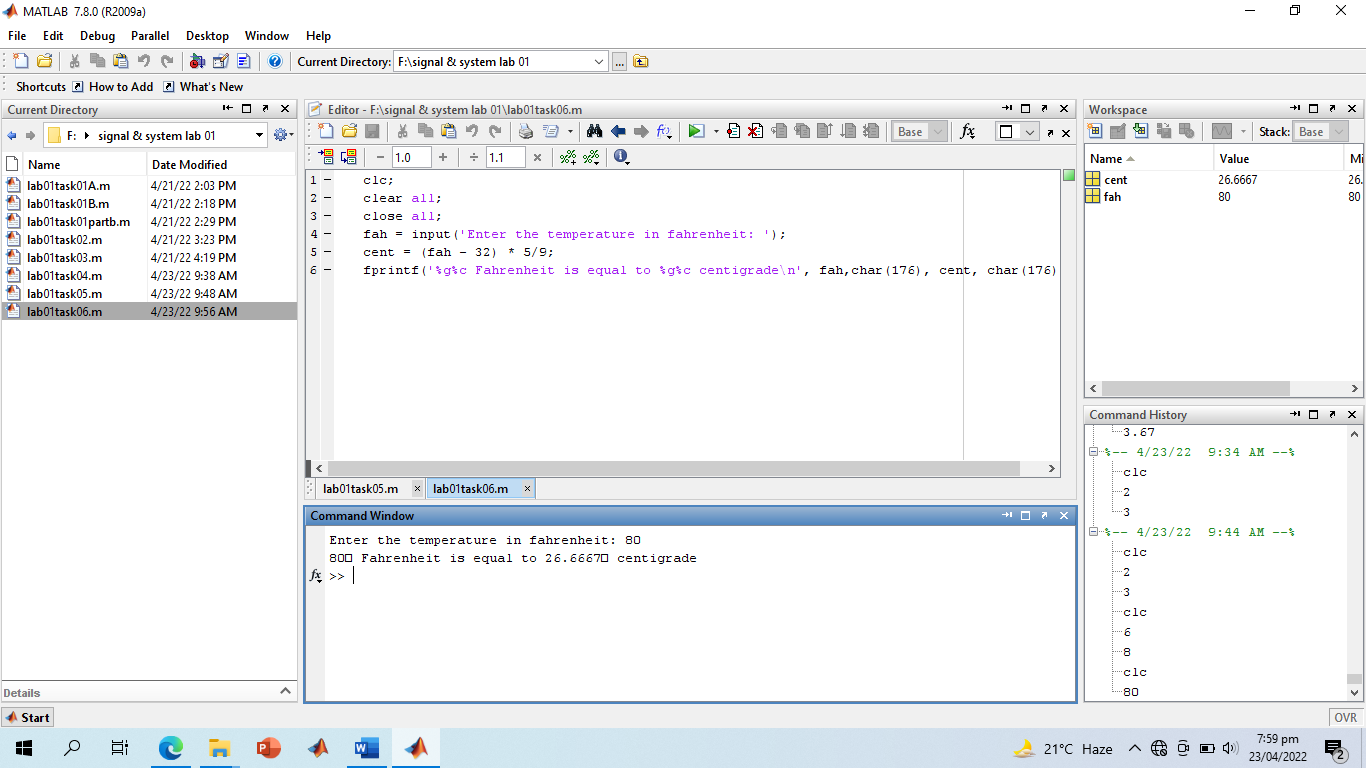
**Algorithm:**

Step 1) Take temperature in Fahrenheit from user.

Step 2) Subtract 32 from the input and multiply the result of subtraction by 5/9.

Step 3) Store the result of multiplication in centigrade. Step 4) Display centigrade.

**Screenshot of Source Code and Output:**



**Discussion and Conclusion:**

We are done with the task but this one definitely requires writing some additional piece of code which quite flexible in that it should allow user to input in unit of his choice. This would definitely require use of if statement and since I am writing this lab report after 2 labs, we haven’t studied if statement. This is why the code I wrote for inter-conversion is very basic.

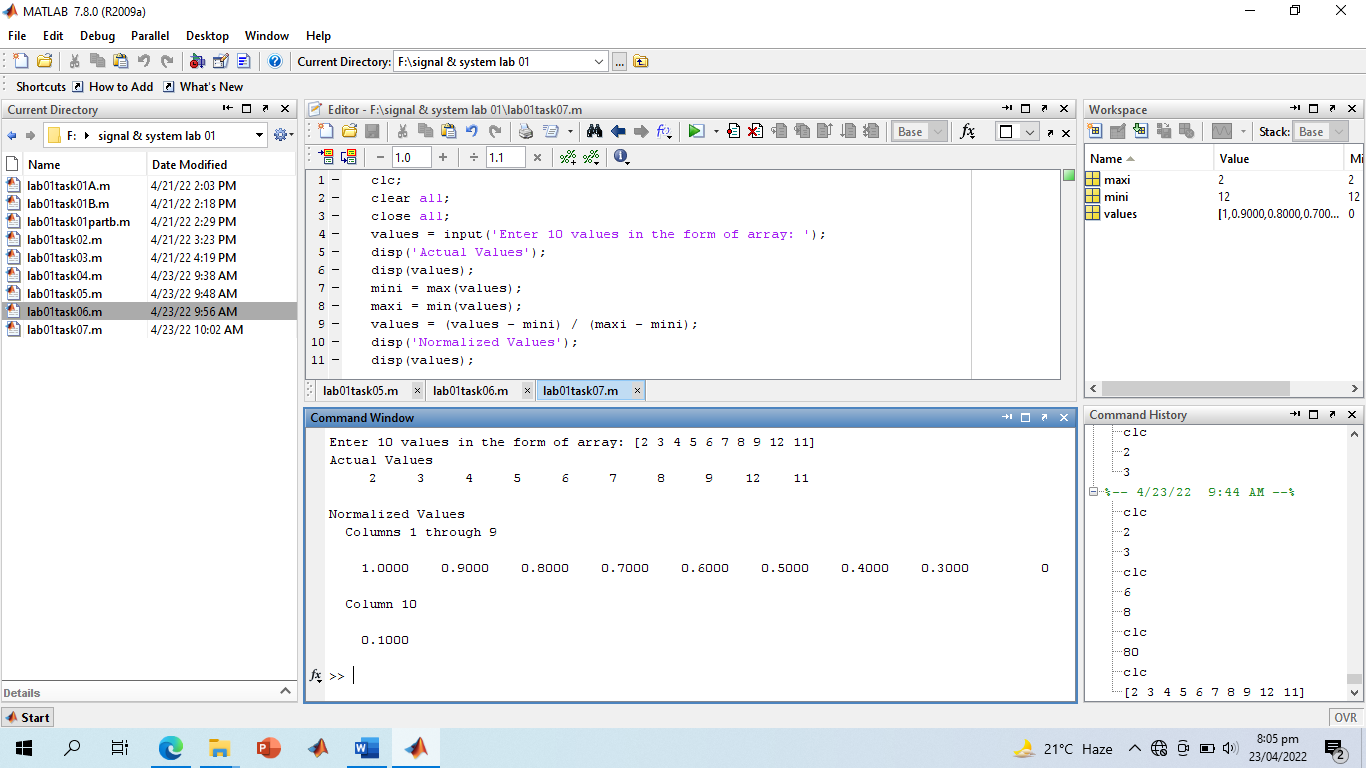
**-------------------------- TASK 07 --------------------------**

Devise an algorithm in MATLAB that takes ten inputs from user and normalize them between [0-1]: Hints: Find the pair wise max of ten numbers using max command. (Variable: mini) Find the pair wise min of ten numbers using min command. (Variable: maxi) Find the normalized value for each input = (input-mini)/(maxi-mini) Note: Do not use loops or if else structures.

**Problem Analysis:**

First up is taking the 10 values from the user, I would have otherwise used a loop for this but since I’m not allowed to use loops for this problem, I will leave it to the user to enter 10 values separated by either space of comma and enclose all values within square braces. One obvious disadvantage with this approach is the fact that user may enter any other number of values than 10. The only solution to this is to check if the array is 10 elements long. This, however, cannot be implemented since using if statement is also discouraged for this problem. So, for input part I would totally have to rely on the user to enter 10 values. After having taken 10 values in an array, I just need to pass the array as argument to max functions and store result in mini variable. Then pass the same array as argument to min function and store result in maxi variable. Then I just need to calculate the normalized values according to the given formula. With this explanation, I don’t think I need to add an Algorithm section for this problem anymore so I’d directly jump to code.

**Screenshot of Source Code and Output:**



**-------------------------- THE END --------------------------**