COMP1200m - Assign 01
Due before 4:45 pm – Friday September 6, 2019
Submit `assign01a.m` and `assign01b.m` via Canvas

Your submitted file name(s) must be spelled and cased as instructed. The file extension must be correct, also.
MATLAB adds the `.m`

Individual submission
NO groups
NO collaboration statement

❈ ❈

Computer programming is not a spectators sport!

Learning to use a programming language requires time and hands on experience.

You will benefit greatly from typing examples from the text and class discussion into MATLAB and observing how they work.

Mistakes are not a bad thing. You will benefit from making mistakes and correcting them.

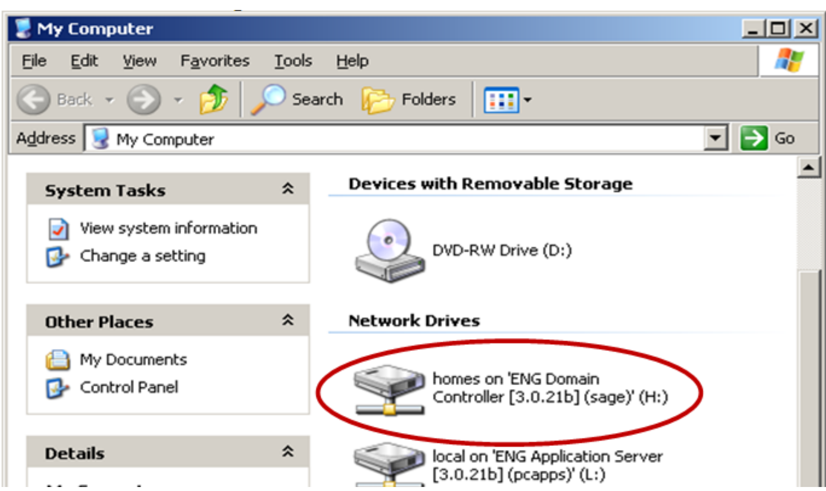
❈ ❈

This assign can be done

- In one of the College of Engineering (EOC) computer assigns.
  - You will need an Engineering account.
- Or on your own personal computer
  - You will need to purchase the software.
- **No group submission for Assign01**.
  - Each student will submit the two files listed above.
- **No collaboration statements are needed in the Assign01 files.**

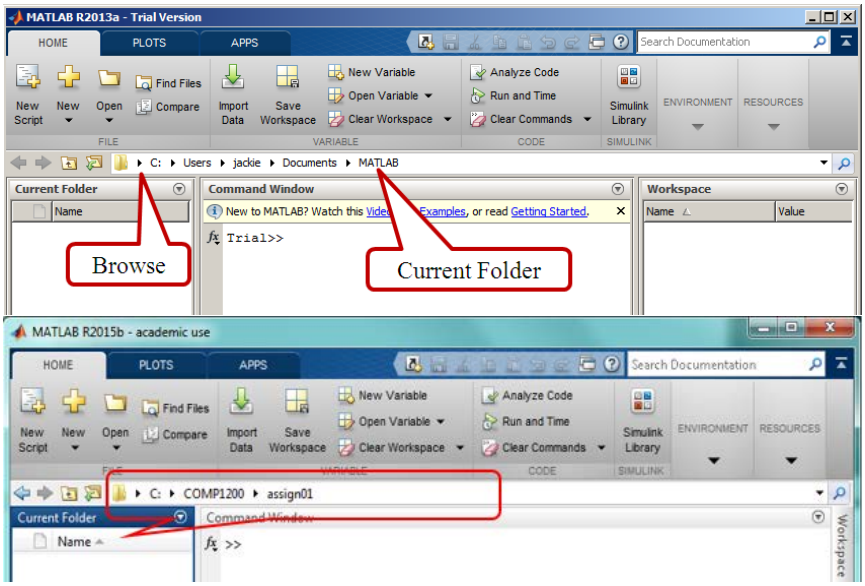
Create a COMP1200 folder to hold your MATLAB files. <<<<<

- If you will be working on an EOC computer, create the **COMP1200** folder on your H: drive. Otherwise, create one on your own C: drive or USB drive.
- Inside your **COMP1200** folder, create two folders: **Notes** and **Assignments.**
- Inside the **Assignments** folder, create a **assign01** subfolder. Many of your assignments will have more than one file; you will need a subfolder for each assignment.
-



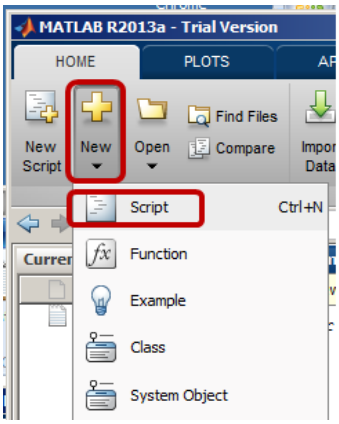
PART A: Start MATLAB.

- You should see a screen similar to the one below.
- Look at the CURRENT FOLDER in the text box on the MATLAB toolbar. Using the arrows, BROWSE to find your **assign01** folder inside your **COMP1200** folder. <<<<<

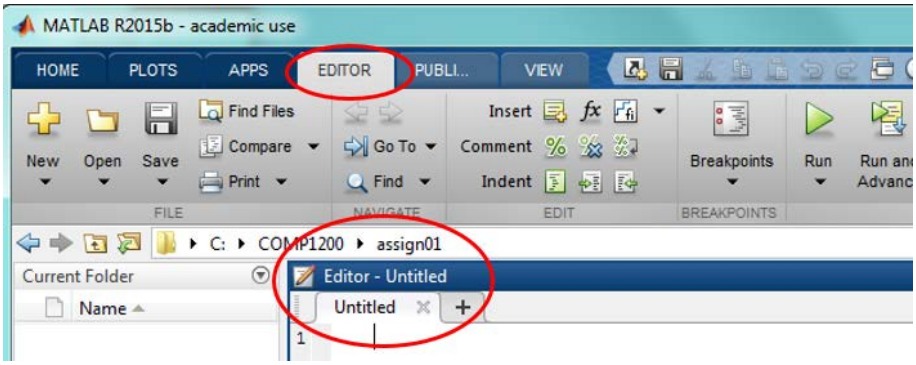


Create an m-script file with the file name **`assign01a.m`**

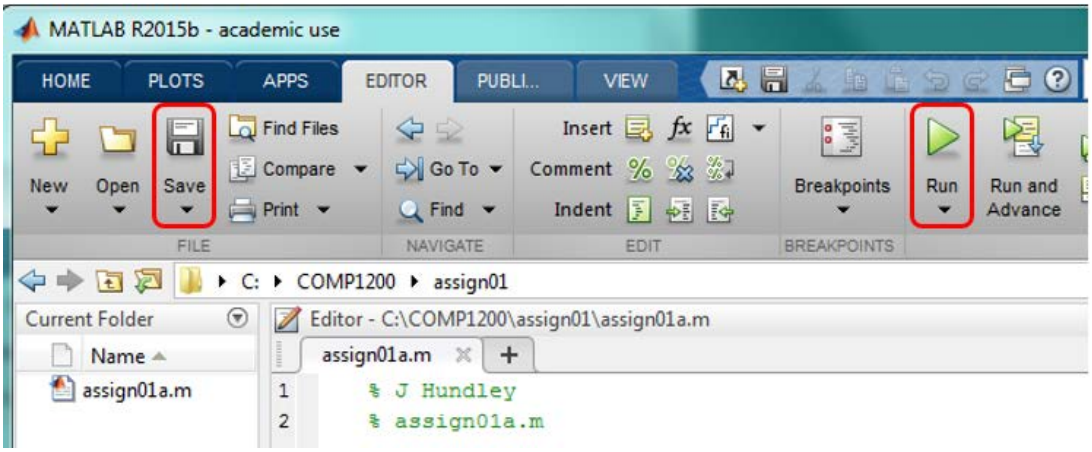
- An m-script file is a MATLAB program file. It allows you to type and save MATLAB statements AND rerun and/or edit them at a later time.
- Use one of the following to open a blank m-script file:
  - Click on the **New Script icon**. See above.
  - Use **New / Script** on the menu bar. Seen here.
  - **Ctrl+N**



- The **Editor Window** appears above the **Command Window** where MATLAB statements are typed.
- Use **File / Save As…** to save **assign01a.m** into the **Current Folder**. The m-script file will have an **.m** extension. **Untitled** will be replaced with the path to **assign01a.m**.



- In the **Editor Window**, start typing the program found on the next page.
- Click the **Save icon** often to prevent loss of work. Do not rely on Autosave (.asv)
- After the statements are typed, click the **Run icon**. **Errors** will be listed in the **Command Window** with line numbers.
- Compare your line with the given line. Look closely.



Where there are **CAPS** type the requested information. Enter the rest of the program below **exactly as is**, observing the column restrictions and including the comment and blank lines. The program prints a two line message welcoming you to COMP-1200.

```
assign01a.m
1  %{
2  TYPE YOUR NAME HERE
3  assign01a.m
4  TYPE THE DUE DATE HERE
5  This a simple program which prints a message to
6  welcome you to Auburn.
7  %}
8  clc, clear all
9  format compact
10
11 disp( ['Welcome to COMP-', num2str(60 * (sqrt(400.0)))] );
12 disp( 'War Eagle!' )
```

Yes, you are to type the information between %{ and %} and include the blank lines.

```
Command Window
Welcome to COMP-1200
War Eagle!
>>
```

If you type the program code as given, your output will look like this.

Now let's introduce some errors. Doing the following is not to turn in, but it will give you an idea of how to deal with errors. Everyone will make errors. They can be a teaching tool.

- ☐ 7th line: delete %} and observe the color change. Run and observe the error message.
- ☐ 11th line: delete one of the single quotes and observe the color change. Run and observe the error message.
- ☐ try and observe other changes

Correct any changes you have made before submitting your file.

PART B: Read this before typing the script.

1. Before you start typing the program at the end of these instructions in the MATLAB editor window. You want to think about the design of the problem solution. The program we will write to compute the real roots of a quadratic equation.
To write this program, we will follow the five-step process for problem solving given in your book, p.5:

1. State the problem, clearly.
2. Describe the input values (knowns) and output (unknowns) information.
3. Develop an algorithm, the steps to solving the problem.
   Develop a hand example with sample input and output.
4. Solve the problem by creating a computer program solution.
5. Test the solution with a variety of data sets.

Read through PART B carefully. This is your introduction to a development plan.

2. The first step of the five-step process should come easy in this course, since the problems that are assigned are generally stated clearly to begin with. In general, deciding exactly what the problem is and what you want the computer to do will be relatively difficult.

The problem we are going to solve is to find the real roots for multiple quadratic equations.

3. The second step is to describe the input and the output given by the program. Like the first step, this step will be easier in this course than in the real world.

For our problem, the input/output description is the following.

Constant – NA
Input – coefficients for a quadratic equation
Output - real roots of the quadratic equation

There may be other information is needed to get the output.

Other – determinant, continue flag

4. The third step is to work the problem by hand for a simple set of data. Let's try this for a case with the following information. The test data should test all requirements for solving for the real roots of a quadratic equation.

test data	a	b	c	x2	x1	
	1	3	-10	-5	2	
	25	0	-16	-4/5	4/5	
	6	1	-2	-2/3	1/2	
	0					first coefficient cannot be zero
	1	1	1	D<0		determinant cannot be negative

Now that we have a feel for how to solve the problem, we can develop an algorithm. We will use top-down design meaning that we will start at the top of the problem and break it down into smaller problems that can be solved separately. The first technique of top-down design is problem decomposition, i.e. identification of the pieces of the problem that need to be solved sequentially.

DECOMPOSITION

1.) get the coefficients
2.) compute the real roots
3.) print the results

The second technique of top-down design is stepwise refinement - doing each step of the problem decomposition in enough detail that it can be easily converted to computer instructions. Flowcharts, pseudocode, or several other methods of specifying detail can be used to do the stepwise refinement. We will use pseudocode.

A good program checks for bad input from the user. We will add checks for the requirements when computing the real roots for multiple equations.

STEPWISE REFINEMENT

while there is an equation
1.) get the coefficients so determinant is not negative
   a. get first coefficient not equal to zero
   b. get second and third coefficients
   c. compute determinant
2.) compute real roots
3.) print coefficients and roots

You will see that these steps are comments in the program below and are a guide to what the program is doing.

ALGORITHM

The **ALGORITHM** steps represent an algorithm that will be used to guide you as you write a program. Note, that this is computer language independent.

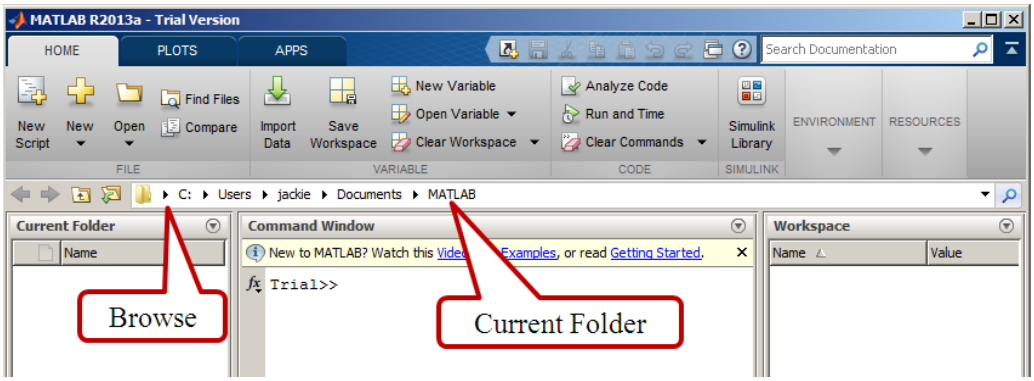
The **ALGORITHM** statements are used as comments in the program.

If needed, the sub-steps from the STEPWISE REFINEMENT can also be used as comments. Further STEPWISE REFINEMENT may be needed to break steps into sub-steps. As the STEPWISE REFINEMENT gets more detailed, the statements may look more like a computer language, but this is NOT a computer programming.

5. Now that the algorithm has been designed, it is time to turn it into MATLAB code. Your MATLAB lectures have not covered all of the computer statements needed to solve this problem. But, if you have followed the design process this far, you will be able to follow what the code is doing This is the second program you must type in and run for this assignment.

START MATLAB.

- You should see a screen similar to the one below.
- Look at the CURRENT FOLDER in the text box on the **MATLAB toolbar**. Using the ARROW, BROWSE to find the **Assign01** folder in your **COMP1200** folder.
- You will see assign01a.m in the **Current Folder window**. Other files that you've saved in your assign01 folder can be seen, also.
- Open a new script in the **Editor Window**.
- Save As your assign01b.m.
- Now, you will see assign01b.m in the **Current Folder window**.
- Type the code on the last page of these instructions.



- When the program runs correctly, enter multiple equations to test ALL PATHS, i.e. does a = 0, is determine < 0, are there more equations. The following is what you will see in the command window during a sample run.

```
Enter 1st coefficient-cannot be zero:
0
Enter 1st coefficient-cannot be zero:
1
Enter 2nd coefficient: 1
Enter 3rd coefficient: 1
*** Determinant is negative. ***
Enter 1st coefficient-cannot be zero:
1
Enter 2nd coefficient: 3
Enter 3rd coefficient: -10

Given: a= 1.0 b= 3.0 c=-10.0
Roots: x1=2.00 x2=-5.00

Do you want to enter another equation?
Enter 1 to continue or 0 to stop: 1
cont =
     1
Enter 1st coefficient-cannot be zero:
6
Enter 2nd coefficient: 1
Enter 3rd coefficient: -2

Given: a= 6.0 b= 1.0 c=-2.0
Roots: x1=0.50 x2=-0.67

Do you want to enter another equation?
Enter 1 to continue or 0 to stop: 0
cont =
     0
```

- While typing and running the statements below, observe…
  - **indenting** and **blank lines** allow the statements to be more readable. (To check indenting, right click in the editor window. Use **Select all** and **Smart indent** in the menu.)
  - **comments** are a guide to what is going on
  - the **colors** have meaning
  - `\n` is "new line" and prints on the next line
  - `clc` clears the Command Window

Submit via Canvas:

`assign01a.m` .m script file, a MATLAB program
`assign01b.m`

Your submitted file names(s) must be spelled and cased as instructed. The file extension must be correct, also.
MATLAB adds the `.m`

Individual submission
NO groups
NO collaboration statement
ONLY the comments given below

Where there are **CAPS**, type the requested information. Enter the rest of the program below **exactly as is**, observing the comments, indenting, and blank lines.

- Don't forget the semicolons to prevent assignments statements from printing the values.
- Notice that some words change color as you type. These are **comments**, **string constants**, and **keywords**.

assign01b.m*

```
%{
TYPE YOUR NAME HERE
File name: assign01b.m
COMP1200 - Spring TYPE THE YEAR HERE
Program: Quad Eq Roots - multiple equations
Read the coefficients for a quadratic equation
Compute and print the roots
%}
clc, clear all
format compact

% while there is an equation
cont = 1; % flag says to continue
while cont == 1
    % get the coefficients so determinant is not negative
    det = -1; % force into loop with negative determinant

    while det < 0
        % get first coefficient not equal to zero
        a = 0; % force into loop
        while a == 0
            a = input( 'Enter 1st coefficient-cannot be zero: ' );
        end
        % get second and third coefficients
        b = input( 'Enter 2nd coefficient: ' );
        c = input( 'Enter 3rd coefficient: ' );
        % compute determinant
        det = b^2 - 4 * a * c;
        if det < 0
            disp( '*** Determinant is negative. ***' )
        end
    end % while det<0
    % compute the roots
    x1 = (-b + sqrt( det ) ) / ( 2 * a );
    x2 = (-b - sqrt( det ) ) / ( 2 * a );

    % print coefficients and roots
    fprintf( '\nGiven: a=%4.1f b=%4.1f c=%4.1f \n', a, b, c )
    fprintf( 'Roots: x1=%4.2f x2=%4.2f \n', x1, x2 )

    % is there another equation?
    fprintf( '\nDo you want to enter another equation?\n' )
    cont = input( 'Enter 1 to continue or 0 to stop: ' )
end % while cont
```

Yes, you are to type the information between %{ and %} and include the blank lines.

Yes, you are to type the lines exactly as they are.