**Homework 1**

(Finish by first meeting of Week 2.)

Read: Chapters 1, 2, 3

Learn how to use Matlab and Octave in your first discussion.

Problems: none (you do not need to turn in anything for this homework)

**Homework 2**

(Due by second meeting of Week 2.)

Read: Chapter 4

Problems:

On your own - Do Problem 2.1 on page 52 to make sure you understand the concept of order of operations. You do not need to turn anything in for this problem.

(1) Ask the user for a number **x** as input, then tell the user the value of the following items:
sin(x), cos(x), sqrt(x), abs(x), exp(x), log(x), log10(x)

(2) Allow the user to input the base and height of three triangles and store this information in two arrays (a **base** and **height** array, each containing three elements). Find the area of all three triangles using matrix multiplication. Then tell the user the area of all three triangles.

(3) A cannon is fired on level ground. Allow the user to input the initial total velocity **v0** and initial elevation angle **theta** in degrees (0 degrees = cannon fired horizontally, 90 degrees = cannon fired straight up). Output the **x** and **y** position of the cannon ball from 0 seconds to 1 second in increments of 0.1 seconds in a nice table format with three columns. Your output should look similar to the following table:

|  |  |  |
| --- | --- | --- |
| time | x | y |
| 0 | 0 | 0 |
| 0.1 | 1.21 | 2.28 |
| 0.2 | 2.42 | 4.18 |

(and so on until t = 1.0)
Remember that **theta** should be in radians, not degrees. Assume the initial position is **(x,y)** = (0,0). The vertical acceleration due to gravity is -9.8 m/s2.

(4) Create a table of x, cos(**x**), and sin(**x**) where **x** goes from 0 to 2pi in increments of 0.1pi . The output should be in a three column table, the first column containing the values of **x**, the second column containing the values of cos(**x**), the third column containing the values of sin(**x**).

(5) For the following array of test scores,

[99,58,87,78,100,100,91,93,78,66]

Find the mean, standard deviation, minimum value, maximum value, and total number of scores using Matlab/Octave's built-in functions (such as **mean( )**) and tell the user these values.

Note: In future problems, I will not allow you to use these built-in functions since they are specific to Matlab/Octave (there is no **mean( )** function in Fortran). However, it is important that you know these functions exist.

**Homework 3**

(Due by second meeting of Week 3.)

Read: Chapter 8.1-8.4, begin reading Chapter 8.5

Problems:

(1) Use the following arrays to do problem 1.

a =

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 3 | 5 | 9 |
| 9 | 1 | 1 | 1 |
| 0 | 0 | 10 | 3 |

b =

|  |
| --- |
| 1 |
| 1 |
| 9 |

c =

|  |  |  |  |
| --- | --- | --- | --- |
| 10 | 11 | 12 | 9 |

(1a) Create a 3x1 array **d** that is the sum of **b** and the second column of **a**. The rules for matrix addition are [here.](http://chortle.ccsu.edu/VectorLessons/vmch13/vmch13_9.html)

(1b) Create a 1x4 array **f** that is the sum of **c** and the first row of **a**.

(1c) Create a 3x4 array **g**, where the first element of **b** is added to every element in the first row of **a**, the second element of **b** is added to every element in the second row of **a**, and the third element of **b** is added to every element in the third row of **a**.

(1d) Create a 1x7 array **h**, where the elements of **b** make up the first three elements of **h** and the elements of **c** make up the last four elements of **h**.

(2) The ideal gas law is P = &rho R T, where
P = pressure of gas (Pa)
&rho = mass density, kg/m3
R = specific gas constant for air = 287.05 J/(kg K)
T = temperature (K)

Create a two-dimensional table to determine the density of air at many different values of Pressure and Temperature. Specifically, vary **P** from 50 kPa to 150 kPa in increments of 10 kPa and vary **T** from 250 K to 350 K in increments of 10 K. Vary **P** in the x-direction and vary **T** in the y-direction. Your table should be 11x11. You do not need to include the table headings that list the pressure and temperature.
Hint: The meshgrid( ) command may be helpful.

(3) A teacher has the following spreadsheet:

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Student1** | **Student2** | **Student3** |
| **Quiz1** | 99 | 85 | 67 |
| **Quiz2** | 90 | 65 | 69 |
| **Quiz3** | 98 | 95 | 97 |
| **Quiz4** | 80 | 85 | 89 |
| **Quiz5** | 98 | 77 | 87 |

(3a) Create a 5x1 array called **mean\_quiz\_scores** containing the mean scores from all 5 quizzes.

(3b) Create an 3x1 array called **mean\_student\_scores** containing the mean scores for all three students.

(3c) Determine the mean score for all quizzes and students (the mean of all 15 values in the table) by taking the mean of both **mean\_quiz\_scores** and **mean\_student\_scores**. Did you get the same answer?

(3d) If the average score of Quiz 5 is greater than the average score of Quiz 4, print "Quiz 5 was easy" to the screen.

(3e) There are 15 total quizzes (3 students x 5 quizzes).
- If student 3 had the highest quiz score out of all 15 quizzes, print "Student 3 is my hero" to the screen.
- If he did not have the highest quiz score, print "Student 3 is a zero" to the screen.
(Determine who has the highest individual score using your Matlab/Octave program, don't just read it off the table with your eyes.)

(4) Allow the user to input 3 numbers (A, B, and C). Do the following tasks in this order:

- If A is greater than B, switch the two numbers (the value of A now is stored in B and the value of B now is stored in A).
- If B is the largest number, set it to zero. Otherwise, set it equal to 100.
- If C is a negative number, set it equal to zero.
- Print the 3 numbers to the screen.

(5) Allow the user to input their test score.

- If the score is less than 60, inform the user he is kicked out of the class.
- If the score is greater than or equal to 60 and less than 90, inform the user that he is an average student.
- If the score is 90 or greater, inform the user he is a champion.

(6) Allow the user to input their name. Use switch/case to do the following:

- If the user inputs 'Paul' inform the user that he may be the instructor for MAE-10.
- If the user inputs 'John' or 'Bill' inform the user that he has 4 letters in his name.
- If the user inputs 'Susan' ask the user for a number. If the number is greater than 10, print 'Hello!' to the screen. Otherwise, print 'Bye!' to the screen. You may use an if statement for this part of the problem.
- If the user's name is none of the above, then inform the user that he/she should eat a taco.

**Homework 4**

(Due by first meeting of Week 5.)

Read: Chapter 8.5

Problems: Do not use Matlab/Octave's built-in functions, such as **mean( )** and **std( )**, to do the problems.

(1) Use a **for** loop to calculate the average and standard deviation of the following array:

[112, 222, 33, 44, 55, 66, 77, 88, 99]

Now, use a **while** loop to calculate the average and standard deviation. Check your answer with the **mean( )** and **std( )** functions.

The mathematical formula for the standard deviation can be found [here](http://www.mathworks.com/access/helpdesk/help/techdoc/ref/std.html) (you may use the first or second formula on that webpage).

(2) The Fibonacci sequence is the following:

1, 1, 2, 3, 5, 8, 13, 21, ...

The sequence starts with 1 and 1, then all subsequent values are determined by summing the previous two numbers in the sequence.

Have your program display the first N Fibonacci numbers in a column, where N is a number you select. Choose a value of N that is greater than 2 (do not use the **input** function, simply hard-code the value of N by typing N = some number greater than 2). For example, if you choose N=6 the program should display:
1
1
2
3
5
8

Calculate all Fibonacci numbers except the first two (1, 1) using a **for** loop and display them on the screen. Although you will assign a value to N in your program, make your algorithm general for any value of N > 2.

(3) Make a program that calculates N! where,

N! = N \* (N-1) \* (N-2) \* ... \* 3 \* 2 \* 1

N is a value you choose (as in problem 2, do not use the **input** function, simply state N = some number). For example, if you choose N=5 your program should calculate 120 and display the result to the screen.

Calculate N! using a **for** loop first, then recalculate N! using **while** loop. Did you get the same result? Check your answer with the **factorial( )** function.

(4) Revisit the cannon problem from HW2.

A cannon is fired on level ground. Assume the user inputs an initial velocity (v0) of 100 m/s and an initial elevation angle of theta = 25 degrees. Using a **while** loop, calculate the x and y position of the cannon ball in time steps of 0.1 seconds until the cannon ball hits the ground. Display the time, x-position and y-position in a nice table format with three columns. (Your output should look similar to the table from HW2, problem 3)

Remember that trigonometric functions require input in radians, not degrees. Assume the initial position is (x,y) = (0,0). The vertical acceleration due to gravity is -9.8 m/s2.

Can you think of how to do this problem with a **for** loop? (not required)

**Homework 5**

(Due by first meeting of Week 6.)

Read: Chapter 7.1-7.2, 7.5

Problems: You MAY use Matlab/Octave's built-in functions, such as **mean( )**, to do the problems.

(1) (Problem 7.9 from the book) Use **fprintf( )** and a **for** loop to create a multiplication table from 1 to 13 for the number 6. Your table should look like this:

1 times 6 is 6
2 times 6 is 12
3 times 6 is 18
<4x6 to 12x6 here>
13 times 6 is 78

(2) At 9:23pm PST on February 2nd, 2010, the US dollar was worth:

90.5141202 Japanese yen
00.7159221 Euros
00.6250000 British pounds
01.0569998 Canadian dollars

Create a conversion table of US dollars to the four other currencies listed above. The left column of the table will list the US dollar amount from $10 to $100 in step sizes of $10. The four other columns will list the corresponding amount of yen, euros, pounds, and Candian dollars.

Use the **fprintf( )** command to display the currency values on the screen and provide column headings ("Dollar," "Yen," etc.). All the currencies should be accurate to two decimal places. The first three columns should look like this:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **USD** | **YEN** | **EURO** | **BP** | **CAD** |
| 10.00 | 905.14 | 7.16 | 6.25 | 10.57 |
| 20.00 | 1810.28 | 14.32 | 12.50 | 21.14 |
| 30.00 | 2715.42 | 21.48 | 18.75 | 31.71 |

Write the data in this table in scientific notation to a text file called **currency.txt**.

(3) Download [this text file](http://albeniz.eng.uci.edu/mae10/MAE10_W2011/OCW_Version/OctavilleRainfall.txt) containing monthly rainfall totals for the fictional city of Octaville during 1988-1991. You can download this file by right clicking the link, choosing "Save link as," and saving the file to your computer. "OctavilleRainfall.txt" must be in the same directory as your m-file when you execute your program.

Read in this data using the **fscanf( )** command and answer the following questions:

(3a) Using **fprintf( )** create a two column table displaying the year and the total rainfall for each year accurate to one decimal place.

(3b) Using **fprintf( )** create a two column table displaying the month (1-12) and the average rainfall for each month accurate to one decimal place.

(3c) Which month (1-12) had the highest average rainfall? Which year had the highest average rainfall? Do not read this data off the table. Use Matlab/Octave's built-in functions (or create your own algorithm) to determine this.

**Homework 6**

(Due by second meeting of Week 7.)

Read: Chapter 6

Problems: This assignment consists of three problems, For each problem, you are given a program by me (the "main" program) that calls a function. You must create the function program on your own, but turn in both the main and function programs in the same m-file (you only turn in one file per problem). It is easier for the TA to grade the problems by turning in one file per problem. For each problem you will need to put the following information into each m-file:

clear; % "clear" goes first

your function goes second

copy and paste the "main" program (which I provide) last

(1) Create a function called **intpoly( )** that can numerically integrate the polynomial function y = ax2 + bx + c from a lower limit to an upper limit, where a, b, and c are constants. Your numerical integrator must allow the user to use the [Trapezoidal method](http://people.hofstra.edu/Stefan_Waner/realworld/integral/numint.html#trap) or Riemann sums method (as we used in discussion) depending on the choice of the user. If the value of **choice** is 1, the Riemann sums method will be used to numerically integrate the polynomial (you may use either a left, right, or midpoint Riemann sums method). If the value of **choice** is 2, the Trapezoidal method will be used to numerically integrate the polynomial.

The main program calls the function **intpoly( )** and has the following form (insert the function before this code in your m-file):

% numerically integrate y(x) = ax^2 + bx + c from lowerlimit to upperlimit
a = 2;
b = 2;
c = 1;
% from 0 to 1
lowerlimit = 0;
upperlimit = 1;
% number of discretizations
N = 1000;
% choice: 1 = Riemann, 2 = Trapezoidal
choice = 2;
disp( intpoly(a,b,c,lowerlimit,upperlimit,N,choice) )

The arguments passed down to **intpoly( )** are the polynomial coefficients (a,b,c), the lower and upper limits of integration (lowerlimit,upperlimit), the number of discretizations (N), and the method of integration (choice). The area is returned to the main program.

(2) Create a function called **cannon( )** that calculates the maximum height of a cannon ball's trajectory, the time at which this occurs, and the speed at that time. These three values are returned to the main program, which has the following form (insert your function before this code in the m-file):

x0 = 0;
y0 = 0;
v0 = 20;
angle = 45;
[maxheight,time,speed] = cannon(x0,y0,v0,angle);
fprintf('The max height is %7.2f \n' , maxheight)
fprintf('The time of the max height is %7.2f seconds \n' , time)
fprintf('The speed at the max height is %7.2f \n' , speed)

The arguments passed down to **cannon( )** are the initial x- and y-coordinates (x0,y0), initial speed (v0), and elevation angle (angle).

(3) A magic square of order *n* is an arrangement of *n2* numbers, usually distinct integers, in a square, such that the *n* numbers in all rows, all columns, and both diagonals sum to the same constant. You must construct a function called **magicsq( )** that creates and displays a magic square of order *n*, where *n* is an odd integer. The function will return as output an nxn array to the program that calls it (the "main" program). YOU MAY NOT USE THE BUILT-IN FUNCTION **magic( )** FOR THIS PROBLEM.

The algorithm for creating an odd magic square of order *n* can be found [here](http://en.wikipedia.org/wiki/Siamese_method).

The main program that calls the function program is the following (place your function program above the this code):

n=5;
magicarray = magicsq(n)

**Homework 7**

(Due by first meeting of Week 9.)

Read: Chapter 5

Problems: This assignment consists of three problems, For each problem, you are given a program by me (the "main" program) that calls a function. You must create the function program on your own, but turn in both the main and function programs in the same m-file. IMPORTANT: Place the function before the main program in your m-file.

(1) The [quadratic formula](http://www.purplemath.com/modules/quadform.htm) allows us to find the roots of a quadratic equation of the form,

ax2 + bx + c,

where a, b, and c are constants.

Create a function called **quadform( )** that takes as input the coefficients of the quadratic equation (a,b,c) and returns the two distinct real roots (x1,x2) as output, if they exist. In addition, the function returns a message (flag) that informs the user if an error occurred when trying to find two distinct real roots. The possible errors are: (1) "Only one root is found"; (2) "Imaginary roots are found"; (3) "Any value of x is a solution"; (4) "No zeroes exist." If no errors occur, then flag should be "No errors."

The code should be structured as follows:

clc; clear;

(insert function here)

a = 1;
b = 2;
c = -1;
[x1,x2,flag] = quadform(a,b,c)

(2) A prime number **N** is an integer that is only divisible by 1 and itself (**N**). Create a function called **primey( )** that takes as input an integer **N** and returns as output two items:

(Item 1) The answer to the question, "Is N prime?" If N is a prime number, the function **primey( )** returns 'Yes' to the main program and stores it in a variable called **answer**. If N is not a prime, the function **primey( )** returns 'No' to the main program and stores it in **answer**.

(Item 2) The prime factors of **N** which are stored in the array **primefactors** in the main program. For example, if **N** is 12, the numbers 2, 2, and 3 will be stored in **primefactors**. If **N** is 39, the numbers 3 and 13 will be stored in **primefactors**. If **N** is a prime number, the numbers 1 and **N** will be stored in **primefactors**.

You may NOT use the built-in functions **factor( )** or **isprime( )**.

The code should be structured as follows:

clc; clear;

(insert function here)

N = 15485863;
[answer,primefactors] = primey(N);
fprintf('Is %i a prime? %s \n', N, answer)
fprintf('The primefactors of %i are: \n' , N)
fprintf('%i \n' , primefactors)
N = 12345678;
[answer,primefactors] = primey(N);
fprintf('Is %i a prime? %s \n', N, answer)
fprintf('The primefactors of %i are: \n' , N)
fprintf('%i \n' , primefactors)

(3) One method for finding a root of a function is the [bisection method](http://www2.lv.psu.edu/ojj/courses/cmpsc-201/numerical/bisection.html). Write a function **bisect( )** that finds the root of a polynomial **ax3** + **bx2** + **cx** + **d** over an interval **[A,B]**. The function takes as input the coefficients **a**, **b**, **c**, **d**, and the interval limits **A** and **B**. The function returns as output the value of the root over the interval **[A,B]**, if it exists. The function also returns as output a message (flag) that informs the user if no roots exist over the interval **[A,B]**.

The code should be structured as follows:

clc; clear;

(insert function here)

a = -1;
b = 1;
c = -1;
d = 10;
A = 0;
B = 5;
[root,flag] = bisect(a,b,c,d,A,B)

**Homework 8**

(Due by second meeting of Week 10.)

Read: Chapter 10

Problems:

(1) There are two functions,

w = t - 1
y = t2 + 2

Plot these two functions from t = -1 seconds to t = 1 second with a step size of 0.1 second. The line for w(t) should be in red with circles at the data points. The line for y(t) should be blue with no special marks at the data points. Label the axes and give a title to the figure.

(2) You are a college professor and your students received the following scores on an exam:

78, 98, 77, 88, 43, 56, 76, 81, 19, 90, 79, 84, 86, 99, 100, 100, 94, 91, 34, 54, 65, 66, 63, 34, 59, 60, 67, 92, 84, 79, 61, 64, 61, 79, 87, 82, 91, 90

Plot the scores on a histogram with 5 bins having the ranges 0-20, 20-40, 40-60, 60-80, 80-100.

(3) Triangle Madness:
This problem will utilize almost every concept you have learned in this course. Read the problem statement [here](http://albeniz.eng.uci.edu/mae10/MAE10_W2011/OCW_Version/chaos_homework.htm).