%flowrate.m
%Patrick Utz, 2/2/18, 4.1
%In the metric system, fluid flow is measured in cubic meters per second (m3/s). A cubic foot per second (ft3/s) is equivalent to 0.028 m3/s. Write a script titled flowrate that will prompt the user for flow in cubic meters per second and will print the equivalent flow rate in cubic feet per second. Here is an example of running the script. Your script must produce output in exactly the same format as this:
>> flowrate
Enter the flow in m^3/s: 15.2
A flow rate of 15.200 meters per sec is equivalent to 542.857 feet per sec
%variables: flowMeters = the input value in cubic meters, flowFeet = the output value in cubic feet
%Algorithm: input flow rate in cubic meters per second
convert by multiplying by 0.028
output the flow rate in cubic feet per second
stop

```
clear
flowMeters = input('Enter the flow in m^3/s: ');
flowFeet = flowMeters*0.028;
fprintf('A flow rate of %f meters per sec is equivalent to %f feet per sec \n', flowMeters,
flowFeet)
```

```
>> flowrate
Enter the flow in m^3/s: 5
A flow rate of 5.000000 meters per sec is equivalent to 0.140000 feet per sec
```

%convertRealToInteger.m
%Patrick Utz, 2/2/18, 4.2
%A file "realnums.dat" has been created for use in an experiment. However, it contains real numbers and what is desired instead is integers. Also, the file is not exactly in the correct format; the values are stored column-wise rather than row-wise. Write a script that would read from the given file realnums.dat into a matrix, round the numbers, and write the matrix in the desired format to a new file called "intnums.dat", and run your program with the given data file, show the results of running your script file.
%variables: realnums = matrix containing original matrix from realnums.dat, roundnums = rounded version of realnums with integers only, intnums = row-wise version of roundnums, intnums.dat = data file containing intnums matrix
%Algorithm: input real numbers from "realnums.dat"
convert the matrix to row wise order and round to get integers
output the new matrix to file named "intnums.dat"
stop

```
clear
load realnums.dat;
realnums
roundnums = round(realnums);
roundnums
intnums = roundnums';
intnums
save intnums.dat intnums -ascii
type intnums.dat
```

```
realnums =
   90.5792   27.8498   97.0593
   12.6987   54.6882   95.7167
   91.3376   95.7507   48.5376
   63.2359   96.4889   80.0280
    9.7540   15.7613   14.1886

roundnums =
    91    28    97
    13    55    96
    91    96    49
    63    96    80
    10    16    14
```

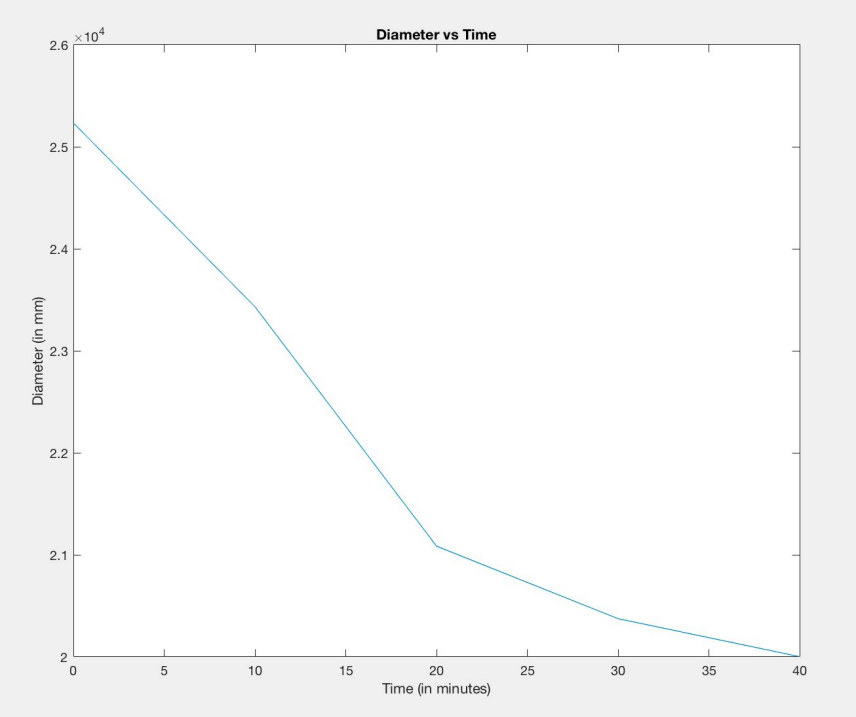
```
intnums =
   91   13   91   63   10
   28   55   96   96   16
   97   96   49   80   14

  9.1000000e+01  1.3000000e+01  9.1000000e+01  6.3000000e+01  1.0000000e+01
  2.8000000e+01  5.5000000e+01  9.6000000e+01  9.6000000e+01  1.6000000e+01
  9.7000000e+01  9.6000000e+01  4.9000000e+01  8.0000000e+01  1.4000000e+01
```

%partDiamPloter.m
%Patrick Utz, 2/2/18, 4.3
%A particular part is being turned on a lathe. The diameter of the part is supposed to be 20,000 mm. The diameter is measured every 10 min and the results are stored in a file called partdiam.dat. Create a data file to simulate this. The file will store the time in minutes and the diameter at each time. Plot the data. Run your program using the attached partdiam.dat as input data file, show the results of running your program.
%variables: partdiam = matrix containing the data from partdiam.dat, x = the time in minutes, y = the diameter in mm
%Algorithm: input time in x vector and respective length in y vector from partdiam.dat
create a plot using the x vector and y vector for x and y axis respectively
output the plot y(x)
stop

```
clear
load partdiam.dat;
x = (partdiam(1:end,1))'
y = (partdiam(1:end,2))'
plot(x,y)
xlabel('Time (in minutes)')
ylabel('Diameter (in mm)')
title('Diameter vs Time')
```

```
>> plotDiamPloter
x =
    0    10    20    30    40
y =
     25233       23432       21085       20374       20002
```



%ExpDecayPlot.m
%Patrick Utz, 2/2/18, 4.4
%Many mathematical models in engineering use the exponential function. The general form of the exponential decay function is: $y(t) = Ae^{-\tau t}$ **where A is the initial value at *t* = 0, and τ is the time constant for the function. Write a script to study the effect of the time constant. To simplify the equation, set A equal to 1. Prompt the user for two different values for the time constant, and for beginning and ending values for the range of a *t* vector. Then, calculate two different y vectors using the above equation and the two time constants, and graph both exponential functions on the same graph within the range the user specified. Use a function to calculate y. Make one plot red. Be sure to label the graph and both axes. What happens to the decay rate as the time constant gets larger? Show the results of running your program.**
%variables: tau1/2 = time constant inputted, t1/2 = range inputted, y1/2 = the two decaying graphs
%Algorithm: input range of time from user and create t vector
input two different values of the time constant (tau) from user
Create y vector with function $y(t) = Ae^{-\tau t}$
output both plots of y(t) for the two tau values on the same graph
stop

```
clear
A = 1;
tau1 = input('Enter first time constant value: ')
tau2 = input('Enter second time constant value: ')
t1 = input('Enter beginning value for time range: ')
t2 = input('Enter ending value for time range: ')
t = t1:.1:t2;
y1 = A*exp(-tau1*t);
y2 = A*exp(-tau2*t);
plot(t,y1, 'r')
hold on;
plot(t,y2, 'b')
title('Exponential Decay For Two Time Constants')
xlabel('Time t (in seconds)')
ylabel('y(t) = Ae^(-tau*t)')
legend('First Time Constant','Second Time Constant')
hold off;
```

```
>> ExpDecayPlot
Enter first time constant value: 2
```

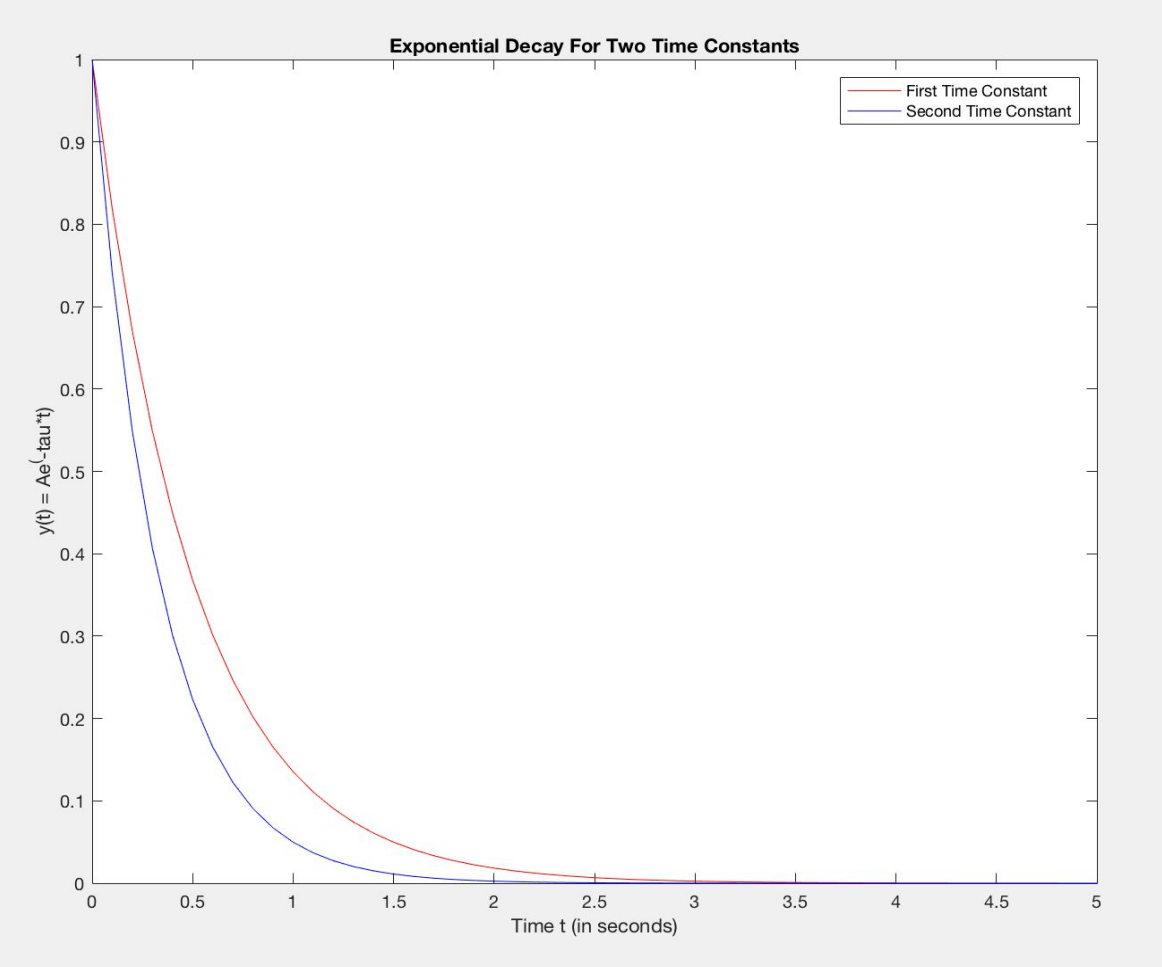
tau1 =
    2

Enter second time constant value: 3
tau2 =
    3

Enter beginning value for time range: 0
t1 =
    0

Enter ending value for time range: 5
t2 =
    5

**The decay rate increases as the time constant gets larger



%FluidPloter.m
%Patrick Utz, 2/2/18, 4.5
%An exponential decaying sinusoid has very interesting properties. In fluid dynamics, for example, the following equation models the wave patterns of a particular liquid when it is perturbed by an external force: y(x) = Fe^(-ax) sin(bx), where F is the magnitude of the external impulse force, and a and b are constants associated with the viscosity and density, respectively. The following data have been collected for the following types of fluids:

Fluid	a value	b value
Ethyl Alcohol	0.246	0.806
Water	0.250	1.000
Oil	0.643	1.213

Write a script that prompts the user for a value for F. Then, create an x-vector with 100 evenly spaced numbers between 0 and 2π, and then calculate the y-vector using the above equation. Plot the wave pattern of a fluid when perturbed for the three different fluids; plot using the values above and compare them when F=5.
%variables: a(fluid name) = a value for respective fluid, b(fluid name) = b value for respective fluid, y(fluid name) = y value for respective fluid, x = time values, F = inputted value for F constant
%Algorithm: input a value for F from the user
create x vector of 100 evenly spaced values from 0 —> 2pi
create three different y vectors of y(x) = Fe^(-ax) sin(bx) using the F value from the user and the a and b values for ethyl alcohol, water, and oil respectively
output the three plots of y(x) for the three liquids on the same graph
stop

```
clear
F = input('Please enter a value for F: ')
aEthyl = 0.246;
aWater = 0.250;
aOil   = 0.643;
bEthyl = 0.806;
bWater = 1.000;
bOil   = 1.213;
x = linspace(0,2*pi,100);
yEthyl = F.*exp(-aEthyl.*x).*sin(bEthyl.*x);
yWater = F.*exp(-aWater.*x).*sin(bWater.*x);
yOil = F.*exp(-aOil.*x).*sin(bOil.*x);
plot(x,yEthyl,'r')
hold on;
plot(x,yWater,'b')
hold on
plot(x,yOil,'g')
xlabel('Time (in seconds)')
ylabel('y(x) = Fe^(-ax) sin(bx)')
```

```
title('Wave Patterns of Different Liquids')
legend('Ethyl Alcohol','Water','Oil')
hold off;
```

```
>> FluidPloter
Please enter a value for F: 5

F =
     5
```