MAE 8 - Winter 2015
Homework 8

Instructions: Follow the homework solution template. Put all answers in a MATLAB script named **hw8.m**. For this homework, you will need to submit multiple files. Create a zip archive named **hw8.zip**. The zip archive should include the following files: **hw8.m**, **read_input.m**, **read_number.m**, **digitsum.m**, **getGCD.m**, **number.txt** and **task1.txt**. Submit **hw8.zip** through TED before 9 PM on 03/6/2015. Use double precision unless otherwise stated.

Problem 1: Download the file **task1.txt** from TED. This text file contains the initial position, velocity and direction for the 4 trajectories in task 1. In this exercise, you will write a function to read these parameters into MATLAB. The function should have the following declaration: **function [Xo, Yo, Zo, Umag, theta, phi, C] = read_input(inputfile, traj_num)** where **inputfile** is a string denoting the name of the file to be read and **traj_num** is an integer indicating the trajectory number. The outputs are the initial position (**Xo**, **Yo**, **Zo**), magnitude of initial velocity (**Umag**), direction (**theta**, **phi**) and coefficient of friction (**C**). The function should have a description. In cases when the input **traj_num** is not available in the file, the function should set all outputs to NaN and display an error warning to screen.

(a) Set **p1a=evalc('help read_input')**;

(b) Read in the file **task1.txt** and get the parameters for trajectory 1. Put the parameters into **p1b**.

(c) Read in the file **task1.txt** and get the parameters for trajectory 2. Put the parameters into **p1c**.

(d) Read in the file **task1.txt** and get the parameters for trajectory 3. Put the parameters into **p1d**.

(e) Read in the file **task1.txt** and get the parameters for trajectory 4. Put the parameters into **p1e**.

(f) Read in the file **task1.txt** and get the parameters for trajectory 5. Put the parameters into **p1f**.

All answers in parts (b-f) should be a 7-element vector listing the parameters in the same order as the outputs in the function **read_input**.

Problem 2: Temperature can be converted from Celsius (°C) and Fahrenheit (°F) to Kelvin (K) by using the following relation:

$$\begin{aligned} T(K) &= T(^\circ C) + 273.15, \\ T(K) &= 5/9 * (T(^\circ F) + 459.67). \end{aligned}$$

(a) Create a 2-element cell array of anonymous functions to define the conversions above. The anonymous function should have a function handle named **K**. The first anonymous function should have the input argument **C** and an expression to convert temperature from

C to **K**. The second anonymous function has the input argument **F** and an expression to covert temperature from **F** to **K**. Set **p2a = K**.

(b) Convert 20° C to Kelvin and put the answer in **p2b**.

(c) Convert 90° F to Kelvin and put the answer in **p2c**.

The ideal gas law is given by: **P V = n R T** where **P** is pressure in atm, **V** is volume in liter, **n** is number of moles, **R** is ideal gas constant and **T** is temperature in Kelvin. For the following exercise, set **R** to be 0.08206 $L\,atm\,mol^{-1}\,K^{-1}$.

(d) Create an anonymous function to relate pressure **P** to other variables in the ideal gas law. The function should have a function handle named **P** and 3 input arguments (**n**, **T**, **V**). Set **p2d = P**.

(e) An 0.12 mol of solid CO_2 is put in an empty sealed 4.0 L container at a temperature of 27°C. When all the solid CO_2 becomes gas, what will be the pressure in the container? Set the answer to **p2e**.

(f) An 0.12 mol of solid CO_2 is put in an empty sealed 4.0 L container at a temperature of 90°F. When all the solid CO_2 becomes gas, what will be the pressure in the container? Set the answer to **p2f**.

(g) Create a figure to show how the pressure changes with temperature in the container with CO_2. Use $n = 0.12\,mol$ and $V = 4.0L$. The x-axis will show the temperature ranging from $25°C$ to $35°C$ with an increment of $0.1°C$ while the y-axis shows the corresponding pressure. Label the axes including units, and give the figure a title. Set **p2g = 'See figure 1'**.

Problem 3: Download the file **number.txt** from TED. The file contains a combination of text and numbers. In this exercise, you will use low-level input functions to read this file into MATLAB. Write a function **get_number** to read in the file. The function should have the following declaration: **function number = get_number(numberfile)** where the input **numberfile** is a string denoting the name of the file and the output **number** is a vector containing the numbers on each line in the file. The function should have a description.

(a) Set **p3a = evalc('help get_number')**.

(b) Set **p3b = get_number('number.txt')** to read in the file and obtain the numbers on each line in the file.

Now write a recursive function **digitsum** to calculate the sum of all digits of a given number. For example, digitsum(312) = 6. The function should have the following declaration: **function dsum = digitsum(n)** where input n is a positive integer and output **dsum** is the sum of all digits in the number **n**. The function should have a description. When the input is not a positive integer, set the output to be a warning message and use **return** to exit the function.

(c) Set **p3c = evalc('help digitsum')**.

(d) Find the digit sum of the first number in the file **number.txt** and set the answer to **p3d**.

(e) Find the digit sum of the second number in the file and set the answer to **p3e**.
(f) Find the digit sum of the third number in the file and set the answer to **p3f**.
(g) Find the digit sum of the fourth number in the file and set the answer to **p3g**.

Now write a recursive function **getGCD** to calculate the greatest common denominator of two given numbers. For example, getGCD(12, 9) = 3. The function should have the following declaration: **function gcd = getGCD(n1, n2)** where inputs **n1** and **n2** are positive integers and output **gcd** is the greatest common denominator of **n1** and **n2**. The function should have a description. When the inputs are not positive integers, set the output to be a warning message and use **return** to exit the function.

(h) Set **p3h = evalc('help getGCD')**.
(i) Find the greatest common denominator for the first and the last number in the file **number.txt** and set the answer to **p3i**.
(j) Find the greatest common denominator for the second and the last number in the file and set the answer to **p3j**.
(k) Find the greatest common denominator for the last number in the file and 525. Set the answer to **p3k**.
(l) Find the greatest common denominator for the last number in the file and 345525. Set the answer to **p3l**.