ENGR131 – Homework 5: Modular Programs (Ch 6) and Strings (Ch 7)

Requirements:

- Submit a single .zip file containing all .m files by the posted due date.
- Every .m file should include a comment at the beginning with your name and network ID.
- All work should be your own, as explained in the Academic Integrity policy from the syllabus.
- Problems will be graded for completeness (reasonable attempt for each rubric item). They will not be checked for correctness.

Using demo programs: Demo programs are provided as .p files. There are three steps for running a .p file: (1) copy it to a folder that's *different* from where you are writing your own program, (2) select that folder in Matlab as the current folder, and (3) type the script name in the Command Window. *WARNING:* Having a .p file in the same folder as your .m file may cause an error message.

Problem #1

Write a program that allows the user to repeatedly analyze the DNA sequence in the file "dna.txt" by counting the number of times a particular pattern occurs. Include a main (primary) function that repeatedly asks for a search pattern until no pattern is entered. Include a subfunction that receives a search pattern and returns the quantity of invalid characters (not a, c, g, or t) in the pattern. If invalid characters are found, print an error message indicating the quantity and do not perform the search.

Sample output: Below are examples of how your output might look:

```
>> problem1
Enter a search pattern: aagct
There are 5 occurrences.
Enter a search pattern: xagz
ERROR: Found 2 invalid characters.
Enter a search pattern: c
There are 818 occurrences.
Enter a search pattern:
```

(At this point, the program has terminated because the user has pressed ENTER without entering a search pattern.)

About DNA: In genetics, a DNA sequence can be represented as a string comprised of four letters: a, c, g, and t (see http://en.wikipedia.org/wiki/Nucleic_acid_sequence). Patterns in a sequence may indicate important features in the genetic code (see http://en.wikipedia.org/wiki/Sequence_motif). For example, the sequence 'gatcctccatatcc' contains the pattern 'tcc' in three places.

Tip (data types): The data file contains the ASCII codes for the characters in the DNA sequence. The following example shows you how to look at the first four values:

```
>> dna = load('dna.txt');
>> dna(1:4)
ans =
    97    99    97   103
```

To use them as characters, the array must be converted to the "char" data type using the "char" function. The following example shows how to convert the first four values (you need to convert the whole array):

```
>> char(dna(1:4))
ans =
acag
```

Now try typing "char(dna)" to see the whole DNA sequence. You may have to scroll over to see all the characters!

Tip (checking for invalid characters): You'll need a loop to check every character in the search pattern. To check a single character, you can use a relational expression. The following examples would check to see if the first character in the string "pattern" is <u>not</u> an 'a'. Notice that the value that is returned (0 or 1) represents false or true.

```
>> pattern = 'aagct';
>> pattern(1) ~= 'a'
ans =
     0
>> pattern = 'xagct';
>> pattern(1) ~= 'a'
ans =
     1
```

A character is invalid if it is not an 'a', not a 'g', not a 'c', and not a 't'. Note that the instructions say to write a subfunction! This will help to organize your program.

Rubric:

Item	**Points**
Has loop to iterate through string	10
Checks results of search	10
Has subfunction for search	10
Total	30

Problem #2

Write a text-based version of the built-in menu function shown in class. Your function will be used with the remaining problems. It should receive a 2-D character array where every row contains the text to be displayed for a single menu item. Each menu item should be displayed beginning with a number for the user to enter in order to select the corresponding item. The function should get the user's selection and return that value. You do *not* need to check for invalid values. Because this function will be used by other programs, it should be saved as a separate .m file. Hint: you can use a loop to print each row of the 2-D array. ADVANCED: Can you display it without using a loop? The following examples show how your output should look:

```
>> textMenu(['Quit    '; 'Option A'; 'Option B'])
1. Quit
2. Option A
3. Option B
Enter your choice: 1
ans =
     1

>> textMenu(['Quit    '; 'Option A'; 'Option B'])
1. Quit
2. Option A
3. Option B
Enter your choice: 2
ans =
     2
```

Rubric:

Item	Points
Has a function	5
Has a loop to iterate through options	5
Total	10

Problem #3

Write a program that allows the user to repeatedly login and logout for an online account using a preset username and password. Your program should contain two functions. The main (primary) function should have a user-input loop that repeatedly allows the user to either quit or change their status (logged in or logged out). Always display a message showing whether the user is logged out or in. Use your function from problem #2 above for the menu. Display an option to either login or logout, but not both. Include a subfunction for logging in that receives nothing and returns a value indicating whether or not the log in was successful. It should ask for a username and password. Use any preset username and password that you wish.

Sample output: Below is an example of how your output might look if the username and password are "chris" and "engr131", respectively.

```
>> problem3
You are currently logged out.
1. Quit
2. Login
Enter your choice: 2
Enter username: chris
Enter password: guess
ERROR: login information does not match our records.

You are currently logged out.
1. Quit
2. Login
Enter your choice: 2
Enter username: chris
Enter password: engr131
Login successful!

You are currently logged in.
1. Quit
2. Logout
Enter your choice: 2

You are currently logged out.
1. Quit
2. Login
Enter your choice: 1
```

Tip (login status): Use a variable to store one value if the user is logged out and another value if logged in. The login function can return one value for success and a different value for failure. This return value can be used by the main function in order to set the user's status (logged out or in).

Rubric:

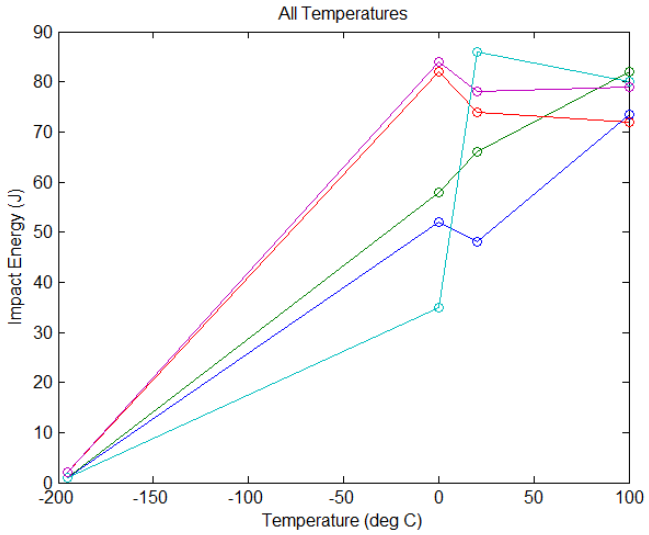
Item	Points
Has a user-input loop	10
Responds to both login and logout options	10
Has subfunction for logging in	10
Total	30

Problem #4

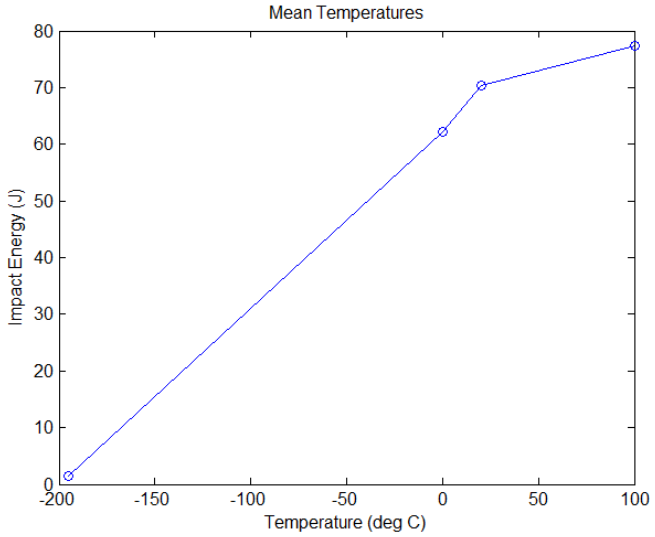
Write a program that creates different graphs for the data file "energyTests.txt". This file contains data from a series of tests for the impact energy absorbed by a metal at various temperatures. In the main (primary) function, use your function from problem #2 above to repeatedly display a menu with options to quit, plot all of the energy levels at each temperature, and plot only the mean energy level for each temperature. Include a subfunction for loading the data file. It should return two arrays: one vector containing just the temperatures (first row of data file) and one matrix containing the remaining rows in the data file. The function should work for data files of any size. Include a subfunction that plots all of the energy values and another subfunction that plots only the mean (average) of the energy values for each temperature.

Sample output: Below is an example of how your output might look with sample graphs.

```
>> problem4
1. Quit
2. All temperatures
3. Mean temperatures
Enter your choice: 2
```




```
1. Quit
2. All temperatures
3. Mean temperatures
Enter your choice: 3
```




```
1. Quit
2. All temperatures
3. Mean temperatures
Enter your choice: 1
```

Data file format: The first row contains four temperatures, and the remaining rows contain the measurements of energy absorption at those temperatures. Each of the remaining rows corresponds to a different metal sample that was tested at all four temperatures.

Tip (computing the mean): Passing a 2-D array as the argument to the built-in "mean" function will produce a vector with one mean value for each column (2nd dimension). For example:

```
>> mean([10 20; 11 21])
ans =
   10.5000   20.5000
```

Tip (so many functions!): Notice that you will have four functions in your program: the main (primary) function, a subfunction to load the data, and two subfunctions for the plotting options.

Tip (plotting): Note that the plot function will automatically use a different color to plot each row of a 2-D array. Here is an example of plotting a 2-D array with three rows: plot([0 1], [10 11; 20 21; 30 31]). Though not required, you can plot lines and circles on the same figure using the plot option 'o-'.

Background: The data for this problem were obtained from the website: http://www.ncsu.edu/labwrite/res/gt/gt-stat-home.html. The impact energy for a material can be measured using the Charpy test (see http://en.wikipedia.org/wiki/Charpy_impact_test). Interested students can watch this video of a Charpy test: http://www.youtube.com/watch?v=a_aOlh6dSA8.

Rubric:

Item	**Points**
Has a user-input loop	10
Has subfunction to load data	5
Has subfunction to plot all temperatures	5
Has subfunction to plot mean temperatures	5
Creates plots	5
Total	30