Name: Andrew Brown	**Lab Time: T 12:00**
People Worked With: Cailin Moore, Kevin Daellenbach, Kyle Magness	**Websites Used:**
Time spent on zyBooks (hrs):2	**Time spent on lab (hrs):5**
Submission Instructions	
Turn all work in to Lab 7 on Gradescope (PDF) and Canvas (.zip file), even if it is not complete yet. If you are not finished, complete the assignment outside of lab and re-submit to Lab *7* on Gradescope and Canvas. All labs are typically due at the same time on Monday every week, but check Canvas if in doubt.	

Learning objectives:

- Create and manipulate strings
- Convert strings to and from numbers
- Reading and writing data files

New MATLAB commands

These are highlighted in **bold** in the instructions below.

- `str = strcat(str1, str2, …)` – glue strings together
- `str = sprintf(" ");` – same as fprintf
- `str = num2str(number);` - convert the number to a string
- `num = str2num(str);` - convert the string to a number
- `[locs] = strfind(str, char);` - find the character in the string
- `bAnswer = isempty(array)` – is the array empty?
- Opening and writing to a file
    - `fid = fopen('filename.txt', 'w')` – open the file (or 'r' for read)
    - `fprintf(fid, '')` – same as fprintf, but writes to a file
    - `fclose(fid)` – close the file

Lab Problems

Files to download to your Lab7 Folder

- *ClassData.csv*
- *ReadData.m*
- *FindCommas.m*

Getting Started

Anything for experimentation goes here with bullet points

- In the command window do:
  - nFound = strfind('hello-bye', '-');
    - What does it return? Count characters
    - What does empty(nFound) return?
  - nFound = strfind('hello-bye', ' ');
  - empty(nFound)
    - What does this return?
- Make sure that [dT, dN] = ReadData('ClassData.csv') works (call it on the command line)
- Open up ClassData.csv in excel
- Match that to what dT and dN have in them

Note: The overall goal is to convert ClassData.csv from its current format to one that looks like lastname_firstname, letter grade (see problem 3). We're going to do this in pieces; Problem 1 and problem 2 create functions that get you set up to do this in Problem 3.

Problem 1

In this problem, you will write two functions. The first function takes in a string and returns that string without any dashes. The second function takes in the first and last name and returns a string that is lastname_firstname and uses the previous function to remove any dashes (-) in the name.
Note that you'll be testing it by calling it from the command line; you'll call it from a script in problem 3.

Deliverables:

1. **Function file** that takes in a string and returns a string without a dash
    a. A script is not necessary for this problem
2. Call that function from the command line and show that it works
    a. strNoDash = RemoveDash('first-name);
        i. For this problem, you will not need to create a script to show your function works.
3. **Function file** that takes in first and last name and returns the joined name
    a. This should include the RemoveDash function ran for both the first and last name
4. Call the function from the command line and show that it works
    a. str = JoinName('first', 'last')
    b. str = JoinName('first, 'last-name);
    c. str = JoinName('first-name, 'last-name);

Step by Step Instructions:

Create a function file that takes in a string and returns the same string with no dashes.

- Add code to your function to check for the dash (-) in either the first or last name
    - Try `strfind`('first'). What do you get?
    - try `strfind`('first-dash'). What do you get?
    - You'll need `isempty` to check for an empty return
- If you have a dash, you need to delete it. There's two ways to do this. Either set that element to the empty array
    - `array(k) = []` - deletes the kth element from array

    - `array = strcat( array(1:k-1), array(k+1:end) )` – set the array to the first and second halves

Create another function file. It should take in two variables and return one.

- The inputs should be the first and last names as strings
- Start with using `strcat` or `sprintf` to join the names together. Check that it works with 'first' and 'last' (call from the command line)
    - Note: One trick is to copy the first line of the function file, delete the function key word, and then put your own variable values in. This makes sure that you have the inputs and the outputs in the right order
    - I.e., copy function function [strOut] = JoinNames(strFirst, strLast) to the command line,
    - And then edit to be
        - strOut = JoinNames('first', 'last')
- Make sure to implement the RemoveDash function for both names in the JoinNames function
    - Call this on both inputs from within your join string function

Self-check:

```
>> strOut = RemoveDash('first-dash')

strOut =

firstdash

>> str = JoinName('first', 'last')

str =

last_first

>> str = JoinName('first-dash', 'last')

str =

last_firstdash

>> str = JoinName('first-dash', 'last-dash')

str =

lastdash_firstdash

>>
```

Grading Criteria:
[25 pts] [Function file for removing dashes]
[25 pts] [Function file for joining names]

[20 pts] [Join names calls RemoveDash]
[10 pts each (3)] [Called functions with inputs given]

Function scripts here:

```
function [name,dashes] = RemoveDash(name)
%REMOVEDASH Takes in a string and returns the same string with no dashes
dashes=strfind(name,'-'); %index where the dashes occur in the char array
name(dashes)=[]; %delete the dashes
end

function [name,dashes] = JoinName(name,last)
%JOINNAME concatenates a first and last name in the form lastname_firstname
%and removes any dashes by calling REMOVEDASHES
name=strcat(last,'_',name); %concatenate the strings
[name,dashes] = RemoveDash(name); %remove dashes
end
```

Command window output

```
>> strOut = RemoveDash('first-dash')

strOut =

    'firstdash'

>> str = JoinName('first', 'last')

str =

    'last_first'

>> str = JoinName('first-dash', 'last')

str =

    'last_firstdash'

>> str = JoinName('first-dash', 'last-dash')

str =

    'lastdash_firstdash'

>>
```

Problem 2

Write a function that takes in a numeric grade and returns a letter grade. 90+ is an A, 80-90 is a B, anything below that is a C.
As before, we'll test this function by calling it from the command window.
Extra credit: Add in +,- for > X7.5 and < X2.5. I.e., B+ is 88.5-90, B- is 80-82.5. Make sure it works for A's and C's. You must do this by looking at the difference, not just a huge number of if-else statements.

Deliverables:

5. **Function file** that takes in numeric grade and returns a string
    a. Once again, you do not need to turn in a script
6. Call the function from the command line
    a. str = ToLetterGrade(95)
    b. str = ToLetterGrade(85)
    c. str = ToLetterGrade(75)
7. Extra credit
    a. str = LetterGrade(98)
    b. str = LetterGrade(82)

Step by Step Instructions:

Create a function file that takes in one input (numerical grade) and returns one output (letter grade)

- There are a variety of ways to create the if statement to create a letter grade output. I'd suggest the following
    - Check if the grade is larger than 90 -> that's an A
    - Check if the grade is larger than 80 -> that's a B
    - Otherwise, a C
- Extra credit: The point is to do this without just using a huge number of if statements. That way, if you were to add in additional letter grades (D and F) or change the boundaries (to 0.3 and 0.7) the code would be easy to change.
    - Calculate the "left over" – e.g. (grade – 80) for a B grade
    - After you assign the main letter grade, add in the '+' or '-' to the current letter, based on the value of the left over. I.e., if the leftover were less than 0.25, you would add a '-' to the 'B' string

Self-check:

```
>> str = LetterGrade(95)

str =
```

```
A

>> str = LetterGrade(85)

str =

B

>> str = LetterGrade(75)

str =

C

>>

EXTRA CREDIT:
>> str = LetterGrade( 98)

str =

A+

>> str = LetterGrade( 82)

str =

B-
```

Grading Criteria:
[40 pts] [Function file that takes in number and returns letter grade]
[20 pts ea] [Correctly assigns A,B,C]
[30 pts EC] Adds in +,-

Function scripts here:

```
function [gradeLetter,leftover] = LetterGrade(grade)
%LETTERGRADE Changes a numeric grade to a letter grade

%Classify grade as A B or C

%if your grade is better than a 70, you have a C
if round(grade)>=70 || round(grade)<70
    gradeLetter0='C'; %make grade C
    leftover=grade-70; %find leftover
end

%if your grade is better than a 80, you have a B
if round(grade)>=80
    gradeLetter0='B'; %make grade B
    leftover=grade-80; %find leftover
end

%if your grade is better than a 70, you have a C
if round(grade)>=90  %if your grade is better than a 70, you have a C
    gradeLetter0='A'; %make grade A
```

```
    leftover=grade-90; %find leftover
end

%Classify grade as +, - or normal

%if your grade exists
if round(leftover)>=0 || round(leftover)<0
    gradeLetter=strcat(gradeLetter0,'-'); %grade has a -
end

%if your grade is better than a -
if round(leftover)>=2.5
    gradeLetter=gradeLetter0; %grade is normal
end

%if your grade is better than normal
if round(leftover)>=7.5
    gradeLetter=strcat(gradeLetter0,'+'); %grade has a +
end
end
```

Command window output

```
>> str = LetterGrade(95)

str =

    'A'

>> str = LetterGrade(85)

str =

    'B'

>> str = LetterGrade(75)

str =

    'C'

>> str = LetterGrade( 98)

str =

    'A+'

>> str = LetterGrade( 82)

str =

    'B-'
```

Problem 3

Read in the ClassData.csv file and write out another csv file that has last_first,LetterGrade. Must use the functions created in problems 1 and 2.

Deliverables:

8. Script to read in data and write out data
9. Output file

Step by Step Instructions:

- Use ReadData.m to read in the data. This function returns a cell array and a numeric array.
    - Check that the data was read correctly by printing out the first name and grade value. Use the command line for this.
    - [dT dN] = ReadData('ClassData.csv')
- Now create a script and read in the data in your script
- Open up a file to write to using `fopen`
    - Look at the ReadData function as an example of how to use `fopen`
- Write out headers for your new data file ('column1Name, column2Name\n')
    - `fprintf`(fid, 'string');
    - Putting the fid in this location prints to the opened file rather than the command window
    - See the self check for how this will look when you open the file
- Loop over all of the names – remember that the first row is a header row, so skip it
    - Use the functions you created to convert the data and create the two output strings
    - Use `fprintf` to print out your new strings to the file (don't forget the comma and the \n)
- Close the file using `fclose` after all of the data has been inputted.

Self-check:

Double click on the output filename in the matlab window. It will bring up a data importer. Grab a screen shot:

ClassDataII.csv

	A Joinedname	B lettergrade
	TEXT	TEXT
1	Joined name	letter grade
2	Summer_Bunny	C-
3	Dutchman_Flying	B
4	deDerain_Flower	A-
5	Ling_Tyvec	C-

Or open up the file in any text editor/excel and copy and paste the data

Grading Criteria:
[10 pts] [Read in the data]
[10 pts] [Open up new file]
[10 pts] [Write out header]
[20 pts] [for loop over names – remember to use size of data here]
[20 pts] [Call functions to convert data]
[10 pts] [Write out new versions]
[10 pts] [Close file]
[10 pts] [Demonstrate it works]

Answer script here:

```
%Andrew Brown Lab 7 Problem 3

clc
clear

%Practice with strings

%Read the given file and store each of its values (use curly brackets to
%read out as strings instead of cells)
[dT,dN] = ReadData('ClassData.csv');
name1=dT{2,1};
name2=dT{3,1};
name3=dT{4,1};
name4=dT{5,1};
last1=dT{2,2};
last2=dT{3,2};
last3=dT{4,2};
last4=dT{5,2};
grade1=dN(2,3);
grade2=dN(3,3);
grade3=dN(4,3);
grade4=dN(5,3);

%Join the first and last names and take away any dashes
[name1] = JoinName(name1,last1); %1st row
```

```
[name2] = JoinName(name2,last2); %2nd row
[name3] = JoinName(name3,last3); %3rd row
[name4] = JoinName(name4,last4); %4th row

%Convert the numerical grades to letter grades
[gradeLetter1,leftover1] = LetterGrade(grade1); %1st row
[gradeLetter2,leftover2] = LetterGrade(grade2); %2nd row
[gradeLetter3,leftover3] = LetterGrade(grade3); %3rd row
[gradeLetter4,leftover4] = LetterGrade(grade4); %4th row

%Create,open,write to, and close a csv file with the new data
fid=fopen('NewClassData.csv','w+'); %create, and open a file to write to
fprintf(fid,'JoinedName, LetterGrade\n'); %Create column names
fprintf(fid,[name1,',',gradeLetter1,'\n']); %row 1
fprintf(fid,[name2,',',gradeLetter2,'\n']); %row 2
fprintf(fid,[name3,',',gradeLetter3,'\n']); %row 3
fprintf(fid,[name4,',',gradeLetter4,'\n']); %row 4
fclose(fid); %close the file
```

Screen shot/copy and paste data file here:

	A	B
	NewClassData	
	JoinedName	LetterGrade
	Text	Text
1	JoinedName	LetterGrade
2	Summer_Bunny	C-
3	Dutchman_Flying	B
4	deDerain_Flower	A-
5	Ling_Tyvec	C-

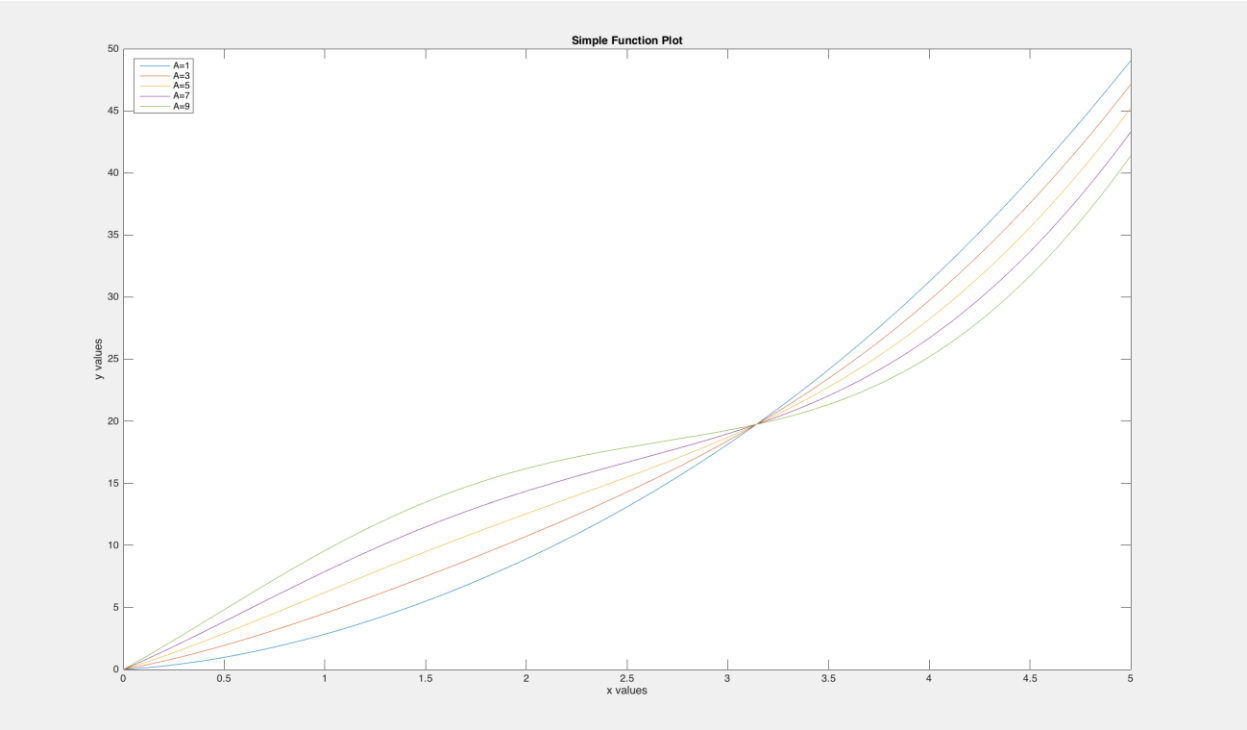
Problem 4- Extra Credit

Plot the function A*sin(x) + 2(x^2) (x = 0 to 5) for 5 values of A. Then, create a legend with string functions. To get the extra credit, an anonymous function and for loop must be used.

Step by Step Instructions:

- The goal of the problem will be to create individual strings for each needed legend entry and store the entries into a cell array, before finally calling legend.
- To start, define A = [1,3,5,7,9]
- Define your function anonymously (Depending on where you place it in your code, you'll need to make your function have 1 or 2 local variables (either @(x) or @(a,x)))
- Plot the function in a for loop while varying the appropriate variable
- Define the legend entry for the loop by stating, as a string, what A equals for the line being plotted. Store into a cell array
- Consider how the previous point should be done. What can be kept the same? What needs to be different? Look up (if you don't already understand) commands like `strcat` or `sprint`. The command `num2str` should also help
- After the for loop, call the legend command with the cell array of legend entries. Move the legend so it doesn't sit on the graphed lines

Self-check:



Grading Criteria:
[10 pts] [Define Anonymous function correctly (based on location within code)]
[10 pts] [Plotted correctly]
[60 pts] [Created legend string in for loop correctly]
[10 pts] [Called legend after loop]
[10 pts] [Moved legend to not cover graph]

Answer script here:

```
% Copy and paste your script here. Make sure your formatting looks the same
as MATLAB, with size 10 font.
```

Plot here:

Command window output

```
Paste output here.
```

zyBooks Challenge Exercises

Do the challenge activities for the following in Week 7

1. Constructing strings
    a. Customized greeting
    b. Markup mania
2. Converting from numbers to strings
    a. Format multiple strings
3. Converting from strings
    a. Times table
4. String parsing and manipulation
    a. Replacing substrings, one fish two fish
    b. Finding a substring: Guest list