Name: Andrew Brown	**Lab Time** *(ECampus write "ECampus")*: T 12:00

Names of people you worked with:
• Kevin Daellenbach

Websites you used:
•

Approximately how many hours did it take you to complete this assignment (to nearest whole number)?	8

By writing or typing your name below you affirm that all of the work contained herein is your own, and was not copied or copied and altered.

Andrew Brown

Note: Failure to sign this page will result in a 50-point penalty. Failure to list people you worked with may result in no grade for this homework. Failure to fill out hours approximation will result in a 10-point penalty.

Turn .zip files to Canvas or your assignment will not be graded

Learning Objectives:

- Attain more practice encapsulating functionality in function files and increasing generality by reading in data from files instead of defining inputs in the script.
- Manipulate strings, turning numbers to strings and vice-versa.
- Working with reading and writing files.

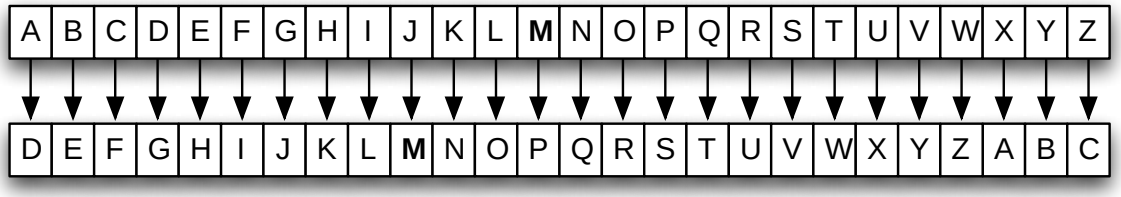
Grading Checkpoints

Criteria	Component	No	Yes
[20%] Comments and Pseudocode	Declared units on all variables?		
	English description of problem at top?		
	Comments outlining your steps?		
[10%] Output formatting	Used fprintf() to make complete sentences (when required)?		
	Correct units on answers?		
	Correct number of decimal places?		
[70%] Functionality	Script computes correct value(s)?		
	Correctly converted units in script when needed?		

We have bolded and italicized primary deliverables for each problem as per request of students to have clearer instructions.

Problem 1

The Caesar Cipher is a simple encryption technique in which all letters are replaced by another letter a fixed number of positions down the alphabet. For example, a **right shift** of **3** on the English alphabet would result in the following:



In this example, the letter M is replaced by the letter P, which is three positions to the right of M. Using this system with a right shift of 3, the string 'MATLABISFUN' would be replaced by 'PDWODELVIXQ'.

If 'A' is encoded as 0, we can use the following equation to shift a single letter:

$$N = (x + s) \,\%\, 26$$

where ***x*** is the numerical encoding of the letter, ***s*** is the Caesar Cipher shift size, and ***N*** is the numerical encoding of the new encrypted letter. (% indicates modulus division… recall the `mod()` function). The same equation can be used to shift every element in a vector. (Note that ASCII encodes A as 65 by default, so you need to shift your starting strings down by 65 so that A = 0 before you can use this equation.)

Write a MATLAB script to do the following:

1. Use `dlmread` to read in the shifts.txt file.
2. Use `dlmread` to read in all of the caesar#.txt files from Canvas inside of a loop.
3. For each "caesar#.txt" file, perform a Caesar Cipher shift according to the corresponding value in "shifts.txt." (E.g. shift the data in caesar1.txt by the number of letters given by the first value in shifts.txt.)
4. ***Print each decoded string (these should be letters, not numbers).***
5. ***Convert each decoded string back to numbers, and then use*** `dlmwrite` ***to write each decoded string to a different text file.*** To build the filename, combine the first 6 characters of the decoded string, the number of the file, and the string '_decoded' to build each filename. For example, the first file (generated from the data in caesar1.txt) should end up as "MATLAB1_decoded.txt." ***Include these files when you turn in your scripts.***

Self-check: (These will all be recognizable English phrases)

```
String 1:
 MATLAB-----
String 2:
 U--T--F-----L--E
String 3:
 T------YG-------OO-----G--------E
```

Comments for grader/additional information (if any)

Script File

```
%Andrew Brown Homework 7 Problem 1

clc
clear

%Solve Caesar Ciphers

text=dlmread('shifts.txt'); %read the caesar shift data
alphabet='ABCDEFGHIJKLMNOPQRSTUVWXYZ'; %index the alphabet

%read, adjust, compute, and print out the plaintext of the given ciphertext
for i=1:3
    if i==1 %using caesar1.txt
        code1=dlmread(strcat('caesar',num2str(i),'.txt')); %store the given
file as a variable
        code1=code1-65; %adjust for ASCII shift
        N1=mod((code1+text(i)),26); %compute proper numbers using given
formula
        string1=alphabet(N1+1); %change numbers to text via indices shifted
b/c A=0
        fprintf(strcat('String 1:\n',string1,'\n')) %print the decrypted
plaintext
    elseif i==2 %using caesar2.txt
        code2=dlmread(strcat('caesar',num2str(i),'.txt')); %store the given
file as a variable
        code2=code2-65; %adjust for ASCII shift
        N2=mod((code2+text(i)),26); %compute proper numbers using given
formula
        string2=alphabet(N2+1); %change numbers to text via indices shifted
b/c A=0
        fprintf(strcat('String 2:\n',string2,'\n'))  %print the decrypted
plaintext
    elseif i==3 %using caesar3.txt
        code3=dlmread(strcat('caesar',num2str(i),'.txt')); %store the given
file as a variable
        code3=code3-65; %adjust for ASCII shift
        N3=mod((code3+text(i)),26); %compute proper numbers using given
formula
        string3=alphabet(N3+1); %change numbers to text via indices shifted
b/c A=0
        fprintf(strcat('String 3:\n',string3,'\n')) %print the decrypted
plaintext
    end
end
```

Function Files (if any)

```
% Copy and paste your functions here. Must be size 10, same as MATLAB font
and color.
```

Command Window Output

```
String 1:
MATLABISFUN
String 2:
USETHEFORCELUKE
String 3:
TOBOLDLYGOWHERENOONEHASGONEBEFORE
>>
```

Problem 2

Epidemic Part 6 – [Conditions to start an epidemic]

Important Deliverables: FindEpidemic function (along with all other functions); plot of function; print out of actual starting non-infected population value.

Extra Credit Deliverables: Printed answers similar to self-check.

Comments for grader/additional information (if any)

Script File

```
%Andrew Brown Homework 7 Problem 2

clc
clear
close all

%Epidemic Part 6 [Conditions to start an epidemic]

%Initial Values
a=10; %the contact rate: the average # of people a person comes in contact
with.
b=1.25; %the amount of time in days that a person is infectious
S0=linspace(0,2000,2000); %Susceptibles, those who have never had the
illness and can catch it.
I0=100; %Infectives, those who are infected and are contagious.
R0=0; % Recovered, those who already had the illness and are immune.
h=0.05; %timestep in days
nSteps=1:140; %timesteps

[S,I,R,N] = DisSimulate(nSteps);

% [bIsEpidemic,peakIValue,peakTime]=IsEpidemic(S0,I0,a,b,nSteps,h);

[mSign,m] = FindEpidemic(I,S0,h); %find the slope of the infected plot

%fit the data and find where it equals 0
P=polyfit(S0,m,20); %convert the data to a polynomial
polyFunc=@(m) polyval(P,m); %solve for the needed values of the new
polynomial P
zero=round(fzero(polyFunc,10)); %Find where the polynomial equals 0
fprintf('Smallest starting population that results in an
epidemic: %0.0f\n',zero) %Print zero

%Plot the data and where the data equals zero
hold on
```

```
plot(S0,m) %plot the starting non-sick pop vs the slope of the infected
plot(zero,m(zero),'xk','Markersize',10); %plot the zero value on the same
graph
axis([0,2000,-100,900]) %define the axes
xlabel('Starting non-sick population') %x label
ylabel('Slope of number of infected') %y label
title('Slope of infection versus starting population') %title of plot
```

Function Files

```
function [mSign,m] = FindEpidemic(I,S0,h)
%FINDEPIDEMIC determies hat starting population is needed to start an
%epidemic.

m=ones(1,length(S0)); %preallocate slope
mSign=ones(1,length(S0)); %preallocate slope sign
for i=1:length(S0) %run the length of S0 times
    m(i)=(I(2,i)-I(1,i))/h; %given formula
    if m(i)<=0 % save the slope sign
        mSign(i)=-1;
    elseif m(i)>0
        mSign(i)=1;
    end
end
end
```

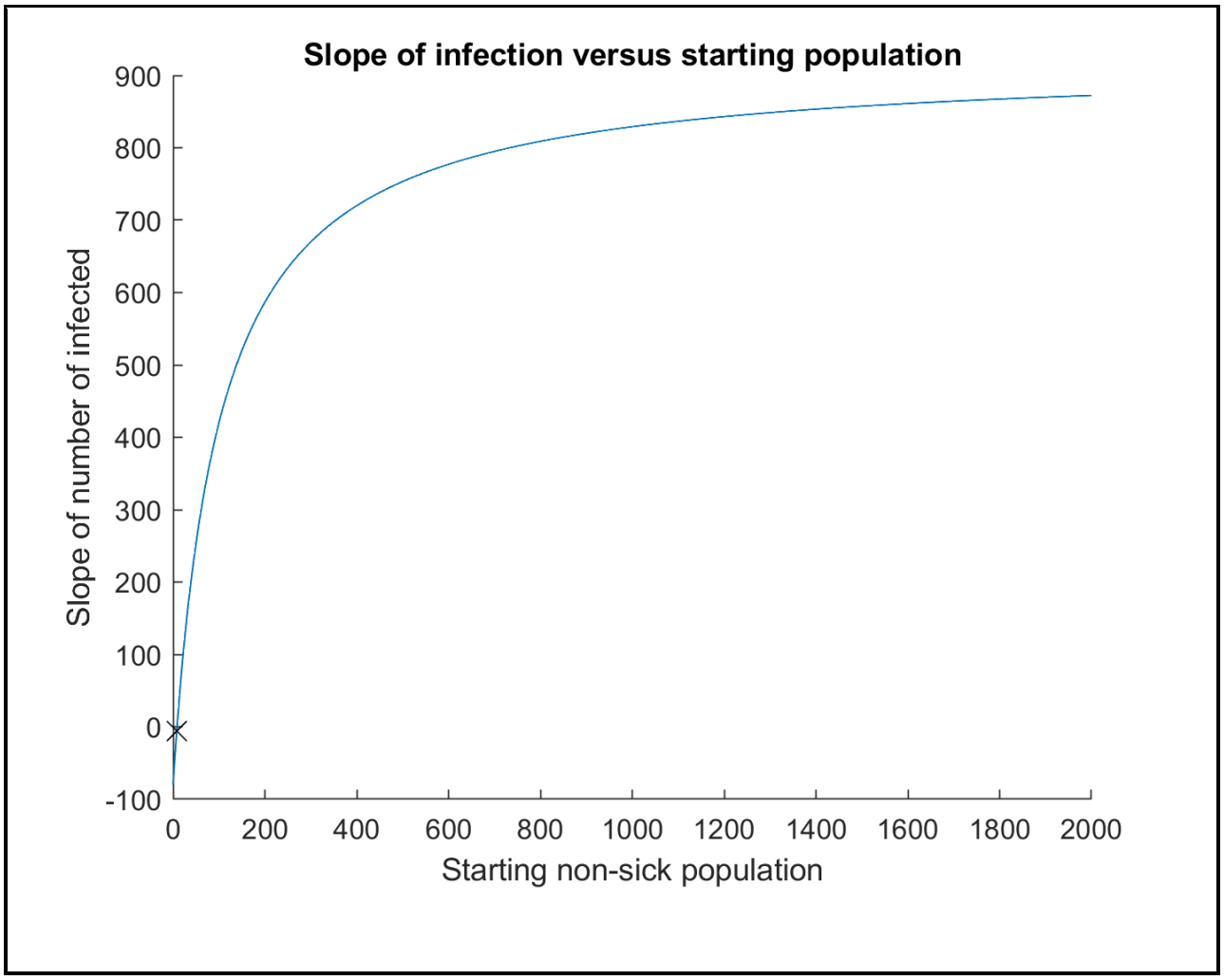
```
function [S,I,R,N] = DisSimulate(nSteps)
%DISEASE SIM:
%   Calls DiseaseStep 140 times with 140 being the length of nSteps
a=10; %the contact rate: the average # of people a person comes in contact
with.
b=1.25; %the amount of time in days that a person is infectious
S0=linspace(0,2000,2000); %Susceptibles, those who have never had the
illness and can catch it.
I0=100; %Infectives, those who are infected and are contagious.
R0=0; % Recovered, those who already had the illness and are immune.
h=0.05; %timestep in days
S(1, :)=S0; %Set first row of S to the S0 array
I(1, :)=I0; %Set the first row of I to all equal the constant I0
R(1, :)=R0; %Set the first row of R to all equal the constant R0

for i=1:(length(nSteps)-1) %run until S, I, and R have 140 values
    [S,I,R,N] = DisStep(S,I,R,S0,I0,R0,a,b,h,nSteps(i)); %call DiseaseStep
and loop through the nSteps array
    S0=S; %Update with new S0
    I0=I; %Update with new I0
    R0=R; %Update with new R0
end

end
```

```
function [S,I,R,N] = DisStep(S,I,R,S0,I0,R0,a,b,h,nSteps)
%DISEASE STEP:
%   Runs one step of he disease and spits out S, I, R, and N

%Set first rows to the given values S0, I0, and R0
if nSteps==1
    S(1, :)=S0; %Set first row of S to the S0 array
    I(1, :)=I0; %Set the first row of I to all equal the constant I0
    R(1, :)=R0; %Set the first row of R to all equal the constant R0
    S=S.*ones(1,length(S0)); %Preallocate S
    I=I.*ones(1,length(S0)); %Preallocate I
    R=R.*ones(1,length(S0)); %Preallocate R
end

%Equations used for calculations of the first timestep.
N=S0+I0+R0; %Total population
dS(nSteps, :)=(-a.*S(nSteps, :).*I(nSteps, :))./N(nSteps, :); %How S changes
dI(nSteps, :)=((a.*S(nSteps, :).*I(nSteps, :))./N(nSteps, :))-
(I(nSteps, :)./b); %How I changes
dR(nSteps, :)=I(nSteps, :)./b; %How R changes

%Calculations for values over time using the previous timestep.
S(nSteps+1, :)=S(nSteps, :)+(h.*dS(nSteps, :)); %# of Sick people after time
h.
I(nSteps+1, :)=I(nSteps, :)+(h.*dI(nSteps, :)); %# of Infected people after
time h.
R(nSteps+1, :)=R(nSteps, :)+(h.*dR(nSteps, :)); %# of Recovered people after
time h.
end
```

Command Window Output

```
Smallest starting population that results in an epidemic: 9
```

Image Output


Slope of infection versus starting population
Slope of number of infected
900
800
700
600
500
400
300
200
100
0
-100
0
200
400
600
800
1000
1200
1400
1600
1800
2000
Starting non-sick population

Problem 3

Total Solar Eclipses (TSEs) Part 4

Comments for grader/additional information (if any)

Answer to question 2:

I got a value of 6585.419 days, while the real value is 6585.3 days. That's a percent error of 0.001807% which is statistically insignificant. However I know the goal of this assignment is to get me to code more, so I will say its "inaccurate" because, yes, it could not have an error and I probably am supposed to complete parts 3 and 4 of this question. The reason for the error is not having enough data points to average and using an average number of days per year and per month instead of the actual varying days per month and per year (leap years) that the Gregorian calendar actually has. I am sure that with more data, or some better date function, I could get to that sweet sweet 0% error.

Script File

```
%Andrew Brown Homwork 7 Problem 3

clc
clear

%Total Solar Eclipse Part 4
%Read the file with the data and consolidate the data by column
data=dlmread('Data.csv'); %read in the file using dlmread
num=data(:,1);
month=data(:,2);
day=data(:,3);
year=data(:,4);
saros=data(:,5);
hour=data(:,6);
minute=data(:,7);
second=data(:,8);
mag=data(:,9);
hour2=data(:,10);
minute2=data(:,11);
second2=data(:,12);

%Use 2 given pairs of saros dates to calculate the length of a saros cycle
[sarosCalc] = SarosCycleCount(second,minute,hour,day,month,year); %call
calculating fucntion
fprintf('SarosCalc = %0.3f\n',sarosCalc) %print my calculated saros cycle

[enhancedSarosCalc] =
EnhancedSarosCycleCalc(second,minute,hour,day,month,year);
fprintf('EnhancedSarosCalc = %0.3f\n',enhancedSarosCalc)

%Calculate the dates of future TSEs using datevec function
firstSarosNum=ones(9,1); %preallocate variable
```

```
secondSarosNum=ones(9,1); %preallocate variable
Y=ones(9,1); %preallocate variable
M=ones(9,1); %preallocate variable
D=ones(9,1); %preallocate variable
for i=3:11 %call the proper indices from the imported file
    firstSarosNum(i-
2)=datenum([year(i),month(i),day(i),hour(i),minute(i),second(i)]); %get the
date numbers of the past TSEs
    secondSarosNum(i-2)=firstSarosNum(i-2)+enhancedSarosCalc; %calculate the
date nums of future TSEs
    [Y(i-2),M(i-2),D(i-2)] = datevec(secondSarosNum(i-2)); %convert date
nums to actual dates
end

%Define numbers for writing
sarosNum=([152 139 126 136 146 133 120 130 145]); %define saros numbers
count=14:23; %define counting numbers

%Print out the dates in the command window
fprintf('Date          Saros\n') %Command window column titles
for i=1:length(sarosNum)

fprintf('%0.0f/%0.0f/%0.0f      %0.0f\n',D(i),M(i),Y(i),sarosNum(i)) %print
end

%Create a matrix out of the calcualted data to write to the data file
matrix=ones(9,5); %preallocate matrix
for i=1:length(sarosNum)
    matrix(i,:)=[count(i),M(i),D(i),Y(i),sarosNum(i)];
end

%Append the data file with the new data calculated
dlmwrite('Data.csv',matrix,'-append','delimiter',',')
```

Function Files

```
function [sarosCalc] = SarosCycleCount(second,minute,hour,day,month,year)
%SAROSCYCLECOUNT calculates the length of a saros cycle in days

%Do the needed year and month conversions into days
daysInASecond=1/86400; %Average number of days in a second
daysInAMinute=1/1440; %Average number of days in a minute
daysInAnHour=1/24; %Average number of days in an hour
daysInAMonth=30.44; %Average number of days in a month
daysInAYear=365.2425; %Average number of days in a year


%Calculate the first saros cycle (saros 127) using 2 given dates
firstsaros127=year(1)*daysInAYear+month(1)*daysInAMonth+day(1)+hour(1)*daysIn
AnHour+minute(1)*daysInAMinute+second(1)*daysInASecond; %day of first date
secondsaros127=year(12)*daysInAYear+month(12)*daysInAMonth+day(12)+hour(12)*d
```

```
aysInAnHour+minute(12)*daysInAMinute+second(12)*daysInASecond; %day of second
date
saros127=secondsaros127-firstsaros127; %days between the 2 dates

%Calculate the second saros cycle (saros 142) using 2 given dates
firstsaros142=year(2)*daysInAYear+month(2)*daysInAMonth+day(2)+hour(2)*daysIn
AnHour+minute(2)*daysInAMinute+second(2)*daysInASecond; %day of first date
secondsaros142=year(13)*daysInAYear+month(13)*daysInAMonth+day(13)+hour(13)*d
aysInAnHour+minute(13)*daysInAMinute+second(13)*daysInASecond; %day of second
date
saros142=secondsaros142-firstsaros142; %days between the 2 dates

%Calculate the average of the two saros cycles
sarosCalc=(saros127+saros142)/2;




function [enhancedSarosCalc] =
EnhancedSarosCycleCalc(second,minute,hour,day,month,year)
%   ENHANCEDSAROSCYCLECLAC Caluclates the length of the saros cycle using
%   special matlab date functions

%Calculate the saros cycle for saros 127
firstsaros127=datenum([year(1),month(1),day(1),hour(1),minute(1),second(1)]);
 %first TSE date
secondsaros127=datenum([year(12),month(12),day(12),hour(12),minute(12),second
(12)]); %second TSE date
saros127=secondsaros127-firstsaros127; %days between the 2 dates

%Calculate the saros cycle for saros 142
firstsaros142=datenum([year(2),month(2),day(2),hour(2),minute(2),second(2)]);
 %first TSE date
secondsaros142=datenum([year(13),month(13),day(13),hour(13),minute(13),second
(13)]); %second TSE date
saros142=secondsaros142-firstsaros142;  %days between the 2 days

%calcualte the average between the 2 saros cycles
enhancedSarosCalc=(saros127+saros142)/2;
end
```

Command Window Output

```
SarosCalc = 6585.419
EnhancedSarosCalc = 6585.334
Date          Saros
4/12/2021      152
8/4/2024      139
12/8/2026      126
2/8/2027      136
```

```
22/7/2028       146
25/11/2030       133
30/3/2033       120
20/3/2034       130
2/9/2035       145
```

Problem 4 [Extra credit]

The Monte Carlo method is used in a wide variety of fields, from finance to engineering to robots to computer science. Named after the famous gambling city of the same name, the method uses random samples to calculate values of functions that are just too difficult to calculate by any other means. It's used in robotics as a way to figure out where the robot is (generate a bunch of possible locations and then keep the most probable ones based on sensor readings). It's used in computer graphics rendering to calculate light diffusing through fog, cracks of doors, or caustics by generating a bunch of light paths bouncing around the scene.

In this problem, you'll use random sampling to do something a bit simpler – calculate the value of pi. You'll simulate throwing darts uniformly at a dart board and calculate how many lie inside the circle versus outside. The ratio of these numbers will tell you an approximation of pi.

Things you will need to know:

- Area of a square with side length 2r: $(2r)^2$
- Area of a circle with radius r: πr^2
- A circle of radius r fits nicely in a square with side length 2r

- The rand function in MATLAB will generate a uniformly random number between 0 and 1
  - Google “uniform random distribution”
- If r is 0.5, the dart board has side length 1…
  - …so put the circle center at (0.5, 0.5)…

To-Do: Calculate the value of pi by throwing 10,000 darts at the dartboard. Repeat these 10,000 throws 400 times and average the result.

a. Show a picture of the dart locations and the dart board for one iteration

b. Print out your answer.

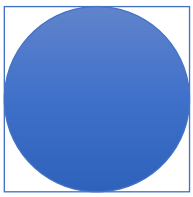
Hints:

1) The ratio of the area of the circle to the area of the square can be used to solve for pi

a. $\frac{Area\ of\ circle}{Area\ of\ square} = A$

b. Rearrange above equation to get pi = (something)

c. “A” is the ratio of the number of darts that end up in the circle versus the number that don’t.



Comments for grader/additional information (if any)

Script File
`% Copy and paste your script here. Must be size 10, same as MATLAB font and color.`

Function Files (if any)
`% Copy and paste your functions here. Must be size 10, same as MATLAB font and color.`

Command Window Output
`Copy and paste the command window output here (same font, size 10).`

Image Output
Copy and paste images here