MATLAB Student Workbook

BME 335

Lesson 1 and 2

Problem 1: Convert each of the decimal numbers to binary (unsigned integer). Try to do this by hand and then use MATLAB (***dec2bin***) to check your answers.

(a) 21
(b) 123
(c) 356

Problem 2: Convert each of the binary numbers (unsigned integers) to decimal. Do this by hand but and use MATLAB (***bin2dec***) to check your answers.

(a) 1010 1101
(b) 0110 1101 0010
(c) 1011 0101 1101

Problem 3:

(a) What is the ASCII string for the word "Computer" in decimal?

Problem 4:

(a) Run the following two commands in MATLAB and record the output of each statement.
>> cos(pi/2) >> cosd(90)
(b) Why do you get two slightly different answers?

Problem 5a: In MATLAB type the following two commands in order to learn about the two functions dec2hex and hex2dec:
>> help dec2hex
>> help hex2dec

(a) Use MATLAB to convert the decimal number 13,465 to hex.
(b) Use MATLAB to convert the hex number 30FF to decimal.

Problem 5b: Try the following commands in MATLAB and explain the output:

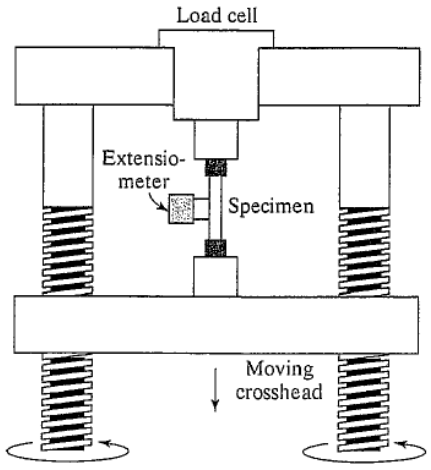
(a) uint8(16.5)
(b) uint8(16.2)
(c) uint8(-47)
(d) uint8(436)
(e) uint16(436)
(f) uint16(1000000)
(g) uint32(1000000)
(h) a = int8(60); 5*a
(i) 5*double(a)

Problem 6: Plotting Functions (Suggestion: Create a script file for this)

Use the subplot command to break the MATLAB figure window into a 2x2 grid of sub-windows. Then plot the following functions using MATLAB commands (**not Plot Tools**) for the formatting. Be sure to make good choices for the values on the x-axis. In all four plots, don't mark the individual data points.

(a) Top Left Sub-window: $y = t^2 - 4*t + 3$ (solid red line - grid)
(b) Top Right Sub-window: $y = \cos(4*t)$ (dashed black line)
(c) Bottom Left Sub-window: $y = e^{-12t}$ (dashed magenta line – grid)
(d) Bottom Right Sub-window: $y = \sin(2\pi t)*\sin(2\pi(20t))$ (solid blue line – use t = 0 to 2 secs)

<u>Problem 7</u>: Tensile Testing (Suggestion: Create a script file for this)
(Modified from William Callister, Materials Science and Engineering, An Introduction, 5th ed.)



A tensile testing machine, like the one shown in the diagram above, is used to apply an increasing force to a material and measure the resulting strain in order to characterize how the material behaves as it deforms.

Applied Force (lbf)	Length after force is applied (in)	Applied Stress, σ (lbf/in^2)	Resulting Strain, ε (in/in)
0	2		
1650	2.003		
3400	2.006		
5200	2.009		
6850	2.012		
7750	2.015		
8150	2.03		
8500	2.06		
8750	2.09		
9000	2.12		

(a) Calculate the *Applied Stress*: σ = F/A and enter the results into the table. F is the applied force (lbf) and A is the cross-sectional area of the material (in^2). Assume the material is a cylindrical rod with a diameter of 0.4 in.

(b) Calculate the resulting *Strain*: ε = (Length – OriginalLength)/OriginalLength. The original length of the rod is 2 inches (i.e., the length with no applied force). Enter the results in the table.

(c) In MATLAB, plot strain on the x-axis and stress on the y-axis. Connect the data points with a solid black line and mark the data points as circles. Label the x and y axis appropriately making sure to include units. Leave your plot open for the next part of the problem.

(d) The point at which the curve begins to flatten is called the yield point or yield stress. If the applied stress is below the yield point, the material will return to its original length when the force is removed. If the applied stress exceeds the yield point, the material has been deformed and will not return to the original shape. Add a text arrow to mark the yield point. To do this, choose Insert in your figure window and select Text Arrow. Place the arrow then type the text (yield point).

<u>Problem 8</u>: Projectile Motion (Suggestion: Create a script file for this)

The x and y position (in meters) of a projectile fired at an initial speed of Vo (m/s), at an angle of θ, and at an initial height above the ground of y_0 (m) are a function of time (s):

$$x_{position} = [V_0 \cos(\theta)]t$$

$$y_{position} = -\frac{1}{2} gt^2 + [V_o \sin(\theta)]t + y_0 \qquad \text{Note: } g = 9.81 \text{ m/s}^2$$

(a) Assume the projectile is fired at 90 m/s and the initial height is 0. On a single graph, plot the y-position of the projectile on the y-axis and time, t, on the x-axis using five different launch angles: 10^o, 25^o, 45^o, 65^o, and 85^o. Your time, t, should range from 0 to 20 seconds in increments of 0.01 seconds. ***Remember: In MATLAB use sind not sin when the angle is in degrees.***

Using MATLAB commands or plot tools or a combination of both, do the following:

- Label the x-axis as Time (s) and the y-axis as Projectile Height (m).
- Adjust the y-axis so the Projectile Height is non-negative.
- Add a legend to show the various launch angles.

(b) Use the Data Cursor Tool to determine the maximum height for each launch angle and the time when the projectile hits the ground for each launch angle. Then calculate the range (final x-position at the time of impact).

Launch Angle (deg)	Maximum Height (m)	Time of Impact (s)	Range (m)
10^o			
25^o			
45^o			
65^o			
85^o			

Lesson 3

Problem 9: Curve Fitting – Trajectory of Projectile

(*You will need the excel files posted on Blackboard: ProjectileData.xlsx . Put this file in your current MATLAB directory)*

The excel file, ProjectileData, has three columns of data: time, distance, and height. Import each of these columns into MATLAB using either xlsread or the Import Data tool. The variable distance represents measurements of the x-position (horizontal position) of the projectile over time and the variable height represents measurements of the y-position (vertical position) of the projectile over time.

The equations for the x and y position of a projectile launched at an angle of θ (rad or degrees) with an initial velocity of V_0 (m/s) were given previously.

(a) Plot **time** on the x-axis and **distance** on the y-axis. Add axis labels (with units) and a title to your plot. We know that the x-position of the projectile increases linearly with time. So use the curve fitting tool to fit a 1st order polynomial (line) to the distance data. Display the equation for the fitted polynomial on your graph with 5 significant digits.

(b) Plot **time** on the x-axis and **height** on the y-axis. Add axis labels (with units) and a title to your plot. We know the height of the projectile follows a parabolic (2nd order) curve. So use the curve fitting tool to fit a 2nd order polynomial (quadratic) to the height data. Display the equation for the fitted polynomial on your graph with 5 significant digits.

(c) Look at the numerical coefficient for the squared term in the fitted polynomial for the **height** data. Theoretically, this coefficient should be equal to -1/2*g. How close is it? Calculate a percent error using the following formula with -1/2*g as actual value:

$$\%Error = \frac{Estimated - Actual}{Actual}$$

(d) The numerical coefficient for the linear term in the fitted polynomial for **height** should be approximately $V_0 \sin(\theta)$ and the numerical coefficient for the linear term in the fitted polynomial for **distance** should be approximately $V_0 \cos(\theta)$. Solve for the initial velocity, V_0, and the launch angle, θ.

Problem 10: Curve Fitting – Decomposition of Hydrogen Peroxide – Estimating Reaction Rates from Experimental Data

(*You will need the excel files posted on Blackboard: HydrogenPeroxide.xlsx. Put this file in your current MATLAB directory)*

First order chemical reactions can be modeled using exponential functions. Hydrogen Peroxide (H_2O_2) decomposes as a 1st order reaction into water and oxygen gas:

$$2H_2O_2 \rightarrow 2H_2O + O_2$$

The concentration of hydrogen peroxide decreases exponentially according to the following

equation:

$$C(t) = C_0 e^{-kt}$$

C(t) = Concentration at time t (M or mols/L) C_0 = Initial Concentration (M)
k = Reaction rate (s^{-1})

The decomposition of hydrogen peroxide in air is a very slow reaction. It decomposes much, much faster in the presence of a catalyst such as Iodide (I^-).

The HydrogenPeroxide.xlsx file has the results of five experiments measuring the concentration of hydrogen peroxide using an Iodide catalyst at 5 different temperatures. Use the curve fitting to estimate the reaction rate, k, for each of these five temperatures.

Take the natural log of the concentration equation, as follows:

$$\ln(C(t)) = -kt + \ln(C_0)$$

- For each of the five temperatures, plot **time** on the x-axis and **ln(C(t))** on the y-axis. Create five separate plots. **Remember, in MATLAB ln(C) is log(C)!**
- Next use the curve fitting tool to fit a line to the data. Display the curve fit equation with five significant digits. The slope of the line should be roughly equal to –k as long as the measurements are good. Enter your estimated reaction rate values in the table below.

Absolute Temperature (K)	Estimated Reaction Rate, k (s^{-1})
280	
285	
290	
295	
300	

<u>Problem 11</u>: Curve Fitting – Decomposition of Hydrogen Peroxide – Estimating Activation Energy from Experimental Data

The reaction rate, k, depends on the temperature according to Arrhenius' Equation (also exponential):

$$k = Ae^{-E_a/(RT)}$$

A = Frequency Factor (s^{-1})
E_a = Activation Energy (J/mol)
R = Ideal Gas Constant = 8.314 (J/(mol*K)) T = Absolute Temperature (K)

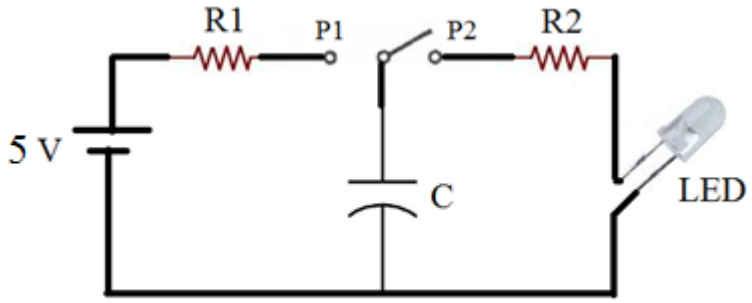
We will again apply the very nice trick of taking the natural log of this equation which gives:

$$\ln(k) = \frac{-E_a}{R} \cdot \left(\frac{1}{T}\right) + \ln(A)$$

Using your five values from the previous problem, plot 1/T on the x-axis and ln(k) on the y-axis. Again, do a linear curve fit. Using the fact that the slope of your line should be approximately $-E_a/R$, determine a value for the activation energy, E_a, for the decomposition of hydrogen peroxide in the presence of an Iodide catalyst.

Problem 12: RC Circuit –Estimating Charging Time

The figure below shows a simple RC (Resistor Capacitor) circuit. When the switch is in position P1, the capacitor will charge at an exponential rate to a voltage equal to the battery voltage (5 V in this circuit). When the switch is in position P2, the capacitor will discharge at an exponential rate to 0 V. While the capacitor is discharging, it will act as a temporary battery to light up the LED (light emitting diode). The time it takes for the capacitor to charge or discharge depends on the size of the resistors and the capacitor, specifically on the product, RC, referred to as a time constant.



Capacitor Charging

The equation that describes how a capacitor charges over time in an RC circuit is:

$$V_c(t) = E(1 - e^{-t/\tau})$$

$V_c(t)$ = Capacitor voltage at time t (V)
E = Voltage of the battery or source (V)
τ is the RC time constant (s)

The time constant, τ, is simply the product of the resistance and the capacitance:

$$\tau = R \cdot C$$

τ = RC time constant (s)
R = Resistance in ohms (Ω)
C = Capacitance in farads (F)

1. Calculate the time constant for each of the resistances shown in the table below.

Note: To calculate τ in seconds, capacitance must be in farads (F) and resistance must be in ohms

(Ω). 1 μF = 10^{-6} F and 1 kΩ = 10^{3} Ω.
Hint: In MATLAB, you can enter the value $4.7*10^{-6}$ as 4.7*10^-6 or as 4.7e-6

Resistance	Capacitance	τ (s)
1 kΩ	4.7μF	
2 kΩ	4.7μF	
10 kΩ	4.7μF	

2. Assume the battery voltage is 5V. On the same graph in MATLAB, plot the capacitor voltage for all three resistance values. Determine a good time range.
3. Add the following to your plot then copy and paste your plot into this document.
   - Label the x-axis as time (s)
   - Label the y-axis as Capacitor Voltage (V)
   - Add a title: Capacitor Charging in RC Circuit
   - Add a legend that shows the three resistance values
4. Using the data cursor tool, determine how long it takes the capacitor voltage to reach 63.2% and 98.2% of its final value (final value is battery voltage). Enter results in the table below:

Resistance	Capacitance	Time to Reach 63.2% of 5V	Time to Reach 98.2% of 5V
1 kΩ	4.7μF		
2 kΩ	4.7μF		
10 kΩ	4.7μF		

Lesson 4

Problem 13: The table below shows the velocity of an object in increments of 0.5 seconds.

t (seconds)	velocity (cm/s)
0	9
0.5	9.25
1.0	10
1.5	11.25

(a) Using the data provided, estimate the velocity of the object at 0.825 seconds using *nearest point* interpolation.
(b) Using the data provided estimate the velocity of the object at 0.825 seconds using *linear* interpolation. Do this part by hand (don't use ***interp1***) and show your calculations.
(c) Suppose we wish to ***insert 3*** new data points (evenly spaced in time, t) between each adjacent set of points in the table. What would the new increment for t be?
(d) Calculate the ***three*** new data points between t = 0.5 and t = 1.0 in the table below using *linear interpolation*. OK to use ***interp1*** for this.

t	velocity (cm/s)
0.5	**9.25**
1.0	**10**

Problem 14: For this problem, you need the excel file, Lesson4_PB14.xlsx posted on the Blackboard. The excel file has a vector of times, Time, which starts at 0 increments by 0.05 and ends at 1 second. It also has a vector of voltage measurements, Voltage, corresponding to the given times. Import both columns into MATLAB using the import tool or the xlsread command.

(a) Use ***interp1*** with a method of ***nearest*** to estimate the voltage every 0.001 seconds between 0 and 1 second. On the same plot (not subplot), plot the original data points as red circles and the interpolated data points as black points.
(b) Use ***interp1*** with a method of ***linear*** to estimate the voltage every 0.001 seconds between 0 and 1 second. On the same plot (not subplot), plot the original data points as red circles and the interpolated data points as black points.
(c) Use ***interp1*** with a method of ***spline*** to estimate the voltage every 0.001 seconds between 0 and 1 second. On the same plot (not subplot), plot the original data points as red circles and the interpolated data points as black points.
(d) What kind of waveform does your plot in part (c) look like? Could you possibly have picked this up from looking at the original data points?

Lesson 5

Problem 15: Do the following in a script file.

(a) Write a set of input statements that prompt the user for the year, month (***a string***), and the day that he/she was born.
(b) Write a single fprintf statement that will display the user's birthday using the variables generated from your input statements.

Problem 16: Do the following in a script file.

(a) Write a set of input statements that prompt the user for his/her favorite band, favorite song, favorite restaurant, and favorite food.
(b) Write two fprintf statements. The first statement should display the user's favorite band and favorite song. The second statement should display the user's favorite restaurant and favorite food on a new line.

Problem 17: Using the data from **Problem 13,** make a script file that contains a menu statement that instructs the user to choose one of the following interpolation schemes: Nearest, Linear, or Spline and saves the user's choice under the variable, Method.
The prompt the user to enter the value that needs to be interpolated. Using the interpolation scheme that was selected, estimate the velocity of the object at given time and display it to the users as message box.
It is always advisable to add checks on the user input to make sure the value is in the required range.

Problem 18: Do this problem by hand in order to practice. Then double check your answer in MATLAB.

(a) What will the two ***disp*** statements produce?

```
a = 5; b = 6; c = 3;
if a > 3 && c < 7 result = a + b*c;
elseif a > 1 && b == 3 result =
        a*c;
elseif b == 5 || c < 5 result = b-15;
end
disp('result = '); disp(result)
```

(b) What will the two ***disp*** statements produce?

```
a = 5; b = 6; c = 3;
if a > 3 && c < 7 result = a + b*c;
end
if a > 1 && b == 3 result = a*c;
end
if b == 5 || c < 5 result = b-
        15;
end
disp('result = '); disp(result)
```

(c) Explain why the code in part (a) produces a different value for result than the code in part (b)

Problem 19: Resistor Value Calculator based on Color Code

Figure 19 shows a picture of a resistor as well as the symbol used for a resistor in a circuit diagram.

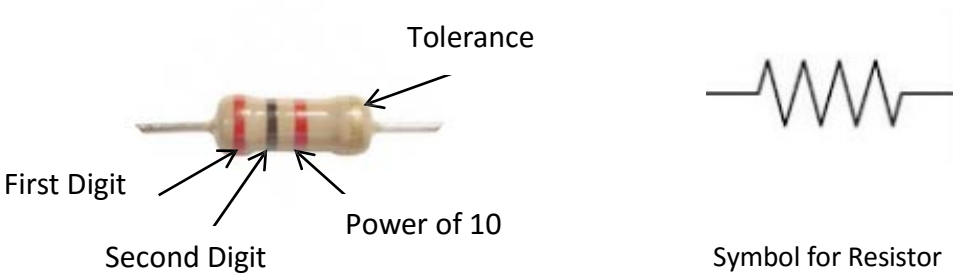


Figure 19: Resistor and Resistor Circuit Symbol

The colored bands on the resistor can be used to identify the resistance. Each color

corresponds to a number as shown in the table below.

Resistor Color Code

Color Bands 1-3	Numerical Value		Color for 4th Band	Tolerance
Black	0		Missing	20%
Brown	1		Silver	10%
Red	2		Gold	5%
Orange	3			
Yellow	4			
Green	5			
Blue	6			
Violet	7			
Gray	8			
White	9			

The nominal value of the resistor can be determined from the color bands on the resistor:

Nominal Value of R = (FirstColorValue*10 + SecondColorValue)*10$^{\text{ThirdColorValue}}$

The tolerance indicates how much the actual value of resistance can vary from the nominal value. The manufacturer's range for the resistor would then be:

Range = Nominal Value ± Tolerance*Nominal Value

Example: Suppose the color bands on the resistor are YELLOW VIOLET ORANGE GOLD

Nominal Value: $(4*10 + 7)*10^3 = 47000\ \Omega = 47\ k\Omega$

Range: $47 \pm 0.05*47\ k\Omega$ or 44.65 to 49.35 kΩ.

(a) Write a script file to do the following:

- Prompt the user for the four colors on the resistor using menu statements
- Calculate the nominal value for the resistor and the range of resistance
- Display (fprintf) the nominal value and range in ohms if the nominal resistance value is smaller than 1000 Ω, in kohms if the nominal resistance value is at least 1000 Ω but less than 1,000,000 Ω and in Mohms if the nominal resistance value is 1,000,000 Ω or higher. Make sure to include units in your fprintf statements. Display two places behind the decimal point for range. Display zero places behind the decimal point if the resistance is in Ω and display one place behind the decimal point if the resistance is in kΩ or MΩ
- Test your script file using the YELLOW VIOLET ORANGE GOLD example from the previous page to make sure your program is working properly.

(b) Run your script for the three cases shown in the table below.

Color Band 1	Color Band 2	Color Band 3	Tolerance Band
Gray	Brown	Black	None
Green	Blue	Red	Silver
Orange	Orange	Blue	Gold

Lesson 7

Problem 20:

(a) Work through the following loop by hand to complete the table below and determine the output of the program. OK to check results using MATLAB but you need to understand how to do this by hand also.

```
x = 0.25; total = 1; for k =
1:4
total = total + (-1)^k*x^k;
end
fprintf('The value of total is:                %0.4f \n',total);
```

	Total
Initial (Before Loop)	
k = 1	
k = 2	
k = 3	
k = 4	

(b) Work through the following loop by hand to complete the table on the next page and determine the program output. Check results using MATLAB but you need to understand how to do this by hand also. Assume the number that the user enters is 95.

```
x = input('Enter a positive integer                ');
product = 1;
N = 1;
while product <= x product =
   product*2; N = N + 1;
end
fprintf('The binary representation of %i requires %i bits
\n',x,N-1);
```

	Product	**N**
Initial		

Problem 21: Cube Root of a Number

The following algorithm can be used to find the cube root of a number:

$$Estimator = \frac{1}{3}(2 * Estimate + Number/Estimate^2)$$
$$Error = abs(Number - Estimate^3)$$

(a) Write a script that will find the cube root of a number using the algorithm shown above with the following specifications:

- Use input statements to prompt the user for the number he/she would like to find the cube root of and for an initial estimate of the cube root.
- Iterate through the algorithm until the Error <= 1e-9
- Use an fprintf statement to display the original number and the estimate for cube root of the number with 3 places behind the decimal point.
- Use an fprintf statement to display the total number of iterations required.

(b) Now run your script to complete the table below:

Number	Estimate	Actual Cube Root	Program Output	Number of Iterations
29.85	**1**			
216	**1**			
2000000	**1**			
2000000	**20**			

Lesson 8

Problem 22: Natural Log

The Taylor Series for the ln(x) for any x in the range $0 < x \leq 2$ is given by:

$$\ln(x) = \sum_{n=1}^{\infty} \frac{(-1)^{n+1}}{n}(x-1)^n = (x-1) - \frac{(x-1)^2}{2} + \frac{(x-1)^3}{3} - \frac{(x-1)^4}{4} + \cdots$$

Obviously, we can't add an infinite number of terms together, but we will use a finite number of terms to get an estimate for ln(x).

1. In order to see how this algorithm works, fill in Table below for x = 1.25.

Iterative Algorithm for ln(x)

Initial	**Estimate = 0**
k = 1	**Estimate = Estimate + $(x-1)^1 / 1$ =**
k = 2	**Estimate = Estimate – $(x-1)^2 / 2$ =**
k = 3	**Estimate = Estimate + $(x-1)^3 / 3$ =**
k = 4	**Estimate = Estimate [illegible] / 4 =**

Note: ln(1.25) = 0.2231. The algorithm provides a pretty accurate estimate after only four iterations for this particular number, x = 1.25.

2. Write a script file to estimate the natural log of a number which is greater than 0 and does not exceed 2 using a finite number of terms from the Taylor Series.
   - Your program should first prompt the user for the number,x, and for the number of desired terms, N.
   - Your program should check and see if x is an invalid number; that is, ***x is less than or equal to 0* or greater than 2**. Use a **while** loop for this! If the number is invalid, prompt the user to enter a new valid value for the number. The **while** loop is nice because it will continue to prompt the user until the user finally enters an acceptable value of x.
   - Your program should then use a **for** loop (**for** k = 1:N) to calculate the estimate of the natural log using N terms. **Remember: in MATLAB, natural log is log.**

Hint: Look at Table above. Each iteration, Estimate = Estimate + New Term. The equation for the New Term changes every iteration. It obviously depends on x. See if you can figure out how to relate the equation to k (the index variable for your loop) also.

   - After the **for** loop, add an fprintf statement to display the estimate of the ln(x) with ***8 places*** behind the decimal point.

3. Test your program to make sure it doesn't accept invalid values for x. Try negative values, zero, and values above 2.

4. Test your program using the values you computed by hand in Table 4. That is, choose x = 1.25 and try 1, 2, 3, and 4 terms.
5. Now use your program to complete Table below.

Test Ln Program Output

x	Number of Terms	Actual Value for ln(x)	Estimate for ln(x)
1.5	1	0.40546511	
1.5	3	0.40546511	
1.5	6	0.40546511	
1.5	7	0.40546511	
1.5	10	0.40546511	
1.5	20	0.40546511	

Lesson 9

Problem 23: Debuging

If you have ever received a letter through the US Postal Service (USPS), you likely saw something like this printed along the edge of the envelope:

This set of dashes is actually a system used by the USPS to encode the zip code to which the letter is being set. The system works in the following way:

1. The set of dashes is broken up into groups of 5, excluding the first and last bars

2. Each set of 5 dashes corresponds to a number based on the pattern (see table below)

Digit	Bar 1	Bar 2	Bar 3	Bar 4	Bar 5
1	0	0	0	1	1
2	0	0	1	0	1
3	0	0	1	1	0
4	0	1	0	0	1
5	0	1	0	1	0
6	0	1	1	0	0
7	1	0	0	0	1
8	1	0	0	1	0
9	1	0	1	0	0
0	1	1	0	0	0

3. The last group (farthest right) is used to check to ensure the zip code is decoded properly; the value of the last group should be the difference between the sum of the values in the zip code and the next largest multiple of 10

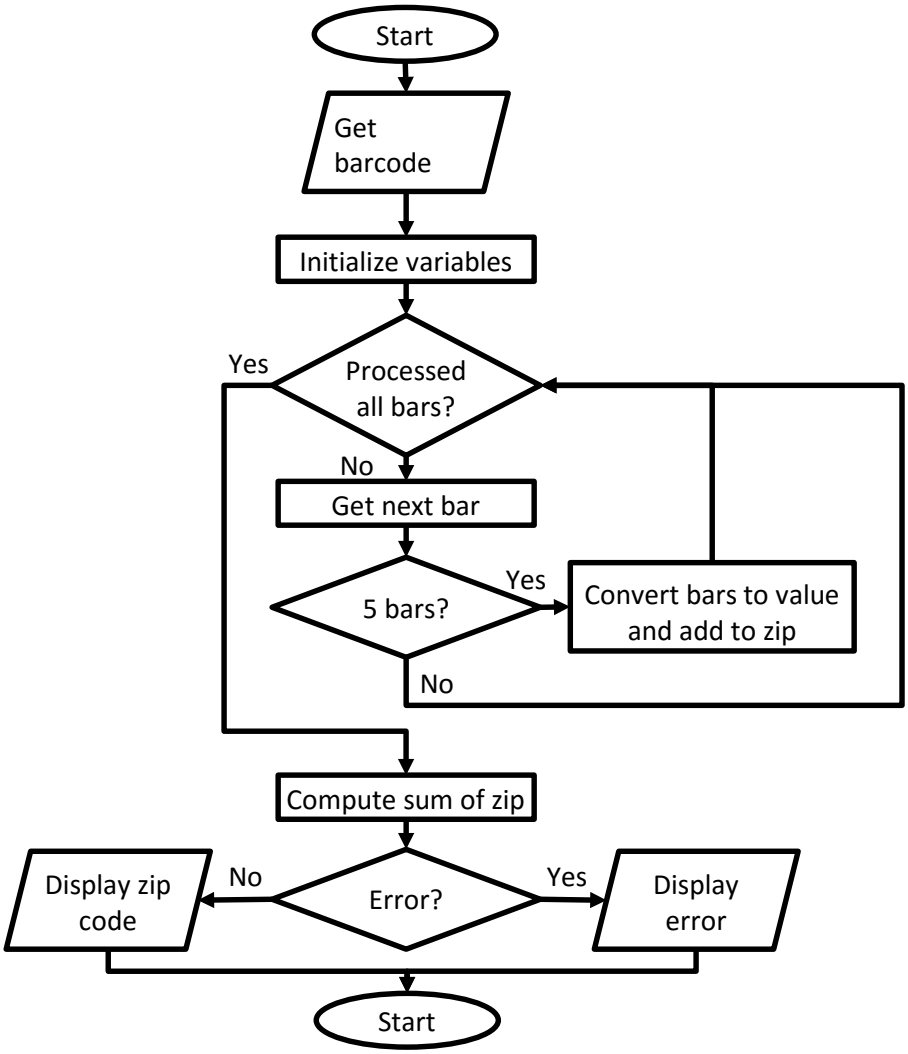
For example, the code above would be translated into:

- II.I...I.I.II.....II.I..I.I.I....II.I..I..II.I..I.I → 95014-5143
- Error check: digits of zip sum to 32, last digit is 8 → 40-32 = 8 → error code checks out

Instructions:

Download the file PB23_Postal_Code.m from Canopy and debug the program so that it will correctly take a string of I's (tall bar) and .'s (short bar) and correctly display the postal code. To help, you have been provided with a flowchart describing how the program is supposed to function, and there are **5** errors in the code.

High Level Flowchart of Postal Code Program



Fill in the table below.

Description of the error you found:	**How you fixed the error:**
1)	
2)	
3)	
4)	
5)	

Lesson 10

Problem 24: Arrays

Download the PB24.mat file and save it in your current MATLAB directory. At the command prompt, type >> load PB24. Look in your workspace window. You should see a 1-d array called ***vector*** (1x12), and a 2-d array called ***matrix*** (10x5).

All of these problems refer to the arrays ***vector*** and ***matrix*** downloaded from the PB24.mat file.

Don't change the values in vector or matrix unless prompted. If you do inadvertently change them, just re-run the command >> load PB24 to recover the original arrays.

Execute the following command first so you know what ***vector*** looks like.

>> vector

a) Give a single MATLAB command that will pull out entries 5, 6, 7, 8, 9 out of the 1-d array, ***vector*** and put them in another vector called ***vectorA***.

b) Give a single MATLAB command that will pull out entries 3, 9, and 11 out of the 1-d array, ***vector*** and put them in another vector called ***vectorB***.

c) Give a single MATLAB command that will overwrite the 3rd entry in ***vectorB*** with a value of 12.

Execute the following command first so you know what ***matrix*** looks like.

>> matrix

a) Give a single MATLAB command that will pull out rows 5, 6, and 7 out of the 2-d array, ***matrix*** and put them in another matrix called ***matrixA***.

b) Give a single MATLAB command that will pull out columns 2, 3, and 4 out of the 2-d array, ***matrix*** and put them in another matrix called ***matrixB***.

c) Give a single MATLAB command that will replace the values in rows 1 & 2 and columns 2 & 3 of ***matrixB*** with the following values:

$$\begin{bmatrix} 42 & 73 \\ -1 & 0 \end{bmatrix}$$

Problem 25: Arrays, Relational Operators, and Useful functions (sum and find)

This problem refers to the array ***vector*** loaded from the PB24.mat file. Again, don't overwrite the values in the array ***vector***. If you do, re-load PB24.mat. For each of these commands, show the result to understand the operation.

(a) vector > 0
(b) vector(3:10) > 0
(c) matrix < 3
(d) 0 < matrix & matrix < 2
(e) matrix(1:3,2:4) > 2

Lesson 12

Problem 26: Arrays

Download the PB24.mat file and save it in your current MATLAB directory. At the command prompt, type >> load PB24. Look in your workspace window. You should see a 1-d array called ***vector*** (1x12), and a 2-d array called ***matrix*** (10x5).
Don't change the values in vector or matrix. If you do inadvertently change them, just re-run the command >> load PB24to recover the original arrays.

Problem 26a: Useful Array Functions (max, min, and sum)

This problem refers to the arrays ***vector*** and ***matrix***, loaded from the PB24.mat file. Again, don't overwrite the values in the arrays ***vector*** and ***matrix***. If you do, re-load PB24.mat.

Execute the following commands first so you know what ***vector*** and ***matrix*** look like.

```
>> vector
>> matrix
```

(a) What does the command: Max = max(vector)do?
(b) What does the command: [Max Loc] = max(vector)do?
(c) What does the command: Max = max(matrix)do?
(d) What does the command: [Max Loc] = max(matrix)do?
(e) What does the command: Max = max(matrix,[],2)do?
(f) What does the command: Max = max(max(matrix))do?
(g) What does the command: Total = sum(vector)do?
(h) What does the command: Total = sum(vector(4:10))do?
(i) What does the command: Total = sum(matrix)do?
(j) What does the command: Total = sum(matrix,2)do?
(k) What does the command: Total = sum(sum(matrix))do?
(l) What does the command: Total = sum(matrix(3:6,4))do?

Problem 26b: Arrays, Relational Operators, and Useful functions (sum and find)

This problem refers to the array ***vector*** loaded from the PB24.mat file. Again, don't overwrite the values in the array ***vector***. If you do, re-load PB24.mat.

(a) sum(vector > 0)
(b) sum(vector > 0 & vector < 2)
(c) sum(vector(1:6)==4)
(d) location = find(vector ==0)
(e) location = find(vector > 0 & vector < 4)
(f) location = find(vector == -4); vector(location) = 173

Problem 27: Arithmetic Operations with Arrays

Determine whether or not the following matrix operations are allowable or not. If the operation is not allowable, indicate this. Otherwise, perform the matrix operations first by hand. Then perform the operations in MATLAB to check your results.

(a) $\begin{bmatrix} -2 & 3 & 8 \\ 5 & 4 & 3 \\ 4 & 5 & 2 \end{bmatrix} + \begin{bmatrix} 1 & -2 & 10 \\ 5 & -6 & 3 \\ 13 & 7 & 8 \end{bmatrix}$

(b) $\begin{bmatrix} -2 & 3 & 8 \\ 5 & 4 & 3 \\ 4 & 5 & 2 \end{bmatrix} - \begin{bmatrix} 1 & -2 & 10 \\ 5 & -6 & 3 \\ 13 & 7 & 8 \end{bmatrix}$

(c) $\begin{bmatrix} 2 \\ -1 \end{bmatrix} \times \begin{bmatrix} 3 & 7 & 6 \end{bmatrix}$

(d) $\begin{bmatrix} -1 \\ 2 \\ 0 \end{bmatrix} \times \begin{bmatrix} -1 \\ 2 \\ 0 \end{bmatrix}$

(e) $\begin{bmatrix} 0 & -2 \\ 4 & 3 \end{bmatrix} \times \begin{bmatrix} 3 & 2 & 0 \\ 6 & -3 & 5 \end{bmatrix}$

Problem 28: Using strcmp and sum

(a) Enter the following commands into MATLAB. Understand the results.

```
>> Resp = {'Yes','No','Yes','No','No','No','Yes'}
>> sum(strcmp(Resp,'Yes'))
>> sum(strcmp(Resp,'No'))
```

Problem 29: Saving Results from Iterative Equations in an Array

Modify your script file from Problem 21 to save all of the estimates in an array then plot the estimates. Note: x-axis values will simply be 1:length(estimate).

Hint: You will need to use indexing to create an array of estimates rather than over-writing the old estimate each time thru the loop – something like this:

$$Estimate(k=1) = {}^{1}/_{3}\,(2 * Estimate(k) + Number/Estimate(k)^2)$$

Run your script for the following four test cases and provide the resulting plot of the estimates for each case.

Number	Estimate
29.85	**1**
-216	**1**
2000000	**1**
2000000	**20**