Practice Debugging Questions

The questions in this packet were written by students in CWRU's ENGR 130 class in Spring 2023 and Autumn 2023. They were selected by the TAs for inclusion in this resource. Titles were added by one of the instructors, but no other content editing has been done. The formatting (indenting, line breaks, etc.) may be slightly different than in the original student submissions, just due to spacing. If you use this resource and have feedback that you'd like to share with us, feel free to send an email to Dr. Harper or slip a note under her office door. (We couldn't ask about this on the course evaluation materials since this is being made available after most of you have filled those out.)

1. **Tylenol Dosages,** created by Anonymous Student

The code is supposed to allow pharmacologists calculate the dosage of Tylenol in a person's body at a given time. It is also supposed to check whether this dosage exceeds the daily threshold of Tylenol that is advised. The code uses the following equation to calculate the daily dosage:

Dosage at a given time = amount of drug taken at the beginning*(0.5 ^(time since the drug was taken/half-life of the drug))

```
%Finding the amount of tylenol present in a body after some time and
%plotting it on a curve
clc;
clear;
close all;

%Asking the user to input values for the amount of tylenol and the amount
%of time that has passed
amt= input ('Please input the amount of tylenol you have consumed today (in mg)');
time = input ('Please input the time that has passed since you took tylenol
(in min)');

%calculating the half life of tylenol
half_life = 2*60;

%calculating the dosage at the user's given time
dosage = (amt) * (0.5 ^ (time/half_life));

%using an if loop to determine whether the dosage is above or below the
%threshold
if dosage <= 4000
   fprintf ('You have exceeded the maximum amount of tylenol for today. Liver
    damage is possible!')
else
   alt_dosage = 4000 - dosage %calculating the dosage that can be taken if its
                              %under the threshold
   fprintf ('You are within the limit - you have %i mg of tylenol in your body
    right now and can consume %s mg of tylenol in the next 24h', dosage,
    alt_dosage)
end

%plotting the time vs dosage in order to visualize the point and the
%corresponding threshold
plot (dosage, time, 'k*');
xlabel ('Time (min)');
ylabel ('Amount of Tylenol (mg)');
yline [4000]; %drawing a line to depict the threshold
ylim([0 10000]); %limiting the y axis so that you can see the threshold and
                 %values
```

2. **National Park Visits**, created by Anonymous Student

The code was written in order to provide the user of the script knowledge about National Park visitation over the years. The code should accomplish asking the user what they want to know about National Parks visitation, then returning the answer to their question.

```
clear;
clc;
close all;

% National Park Data Taken from Website (1904 to 2022)
parkAttendance = [120690 140954 30569 32935 42768 60899 173416 194207 198334
216853 209693 314299 326506 453498 436222 781178 1022091 1101697 1136949
1364024 1527999 1900499 2162640 2465058 2703753 3010912 3038935 3217674
3551885 3255684 6095201 7435659 11749790 14838640 16019483 15141032 16410148
20487633 8891495 6383513 7723790 10855548 20918012 24258527 26294795 29124837
32706172 36613178 41804313 45679754 47967800 48891000 53872100 58220600
58466800 62834000 71586000 78933900 88548300 101959800 109190300 118662500
129282100 135414200 145449500 159103500 168135100 151265400 163156569
166572300 168686500 188085700 215359800 209370600 221127705 205369795
220463211 238592669 244924579 243619396 248785509 263441808 281094850
287244998 282451441 269399837 255581467 267840999 274694549 273120925
268636169 269564307 265796163 275236335 286762265 287130879 285891275
279873926 227299880 266230290 276908337 273488751 272623980 275581547
274852949 285579941 281303769 278939216 282765682 273630895 292800082
307247252 330971689 330882751 318211833 327516619 237064332 297115406
311985998];

% Create a vector of years from 1904 to 2022
years = linspace(1904,2022,118);

% Display options to user and ask for input
option = input('\nWelcome to the National Park Program! \nEnter the number
  corresponding to the action you want to take... \n1) View a graph of
  National Park Attendance \n2) Ask about a specific year \n3) Ask about the
  year with the greatest and least attendance \n');

% Follow the users input
if option == 1                  % The user asked to see a graph
     % Graph park attendance over the years open
     plot(years,parkAttendance);
     hold on;
     xlim(1904 2022)
     ylim([0 360000000])
     xlabel('Park Years')
     ylabel('Yearly Park Attendance')
     title('National Park Visitation over the Years')

     % Add a point to the graph when the parks had the highest visitation
     maxAttendance = max(parkAttendance);
```

xlim takes vector input, should be in brackets ([1904 2022])

when logical indexing, == must be used as a logical operator

```
    maxYear = years(parkAttendance = maxAttendance);
    plot(maxYear, maxAttendance, 'r*');

    % Create a fit to model the increasing attendance of National Parks
    p = polyfit(years, parkAttendance, 2);
    horizontalValues = linspace(min(years), max(years), 1000);
    verticalFit = polyval(p, horizontalValues);
    plot(years, verticalFit);
```

years and vertical fit have different dimensions, plot(horizontalValues, verticalValues)

```
elseif option == 2          % The user wants to ask about a specific year
    % Ask the user what year
    userYear = input('\nWhat year would you like to know about? We have data
      from 1904 to 2022 \n');

    % Display the information for that year
    fprintf('In %i there were %i people who visited the National Parks \n',
      userYear, parkAttendance( years == userYear));

elseif option == 3 % The user asked what year the National Parks had the
                   % greatest attendance
% Find the max attendance and what year that happened
    maxAttendance = max(parkAttendance);
    maxYear = years(parkAttendance == maxAttendance);

% Display info
    fprintf('The greatest yearly visitation of the National Parks was
    maxAttendance people in maxYear');
```

maxAttendance, maxYear)

%i %i

```
end
```

fprintf only works if variables are specified outside of message

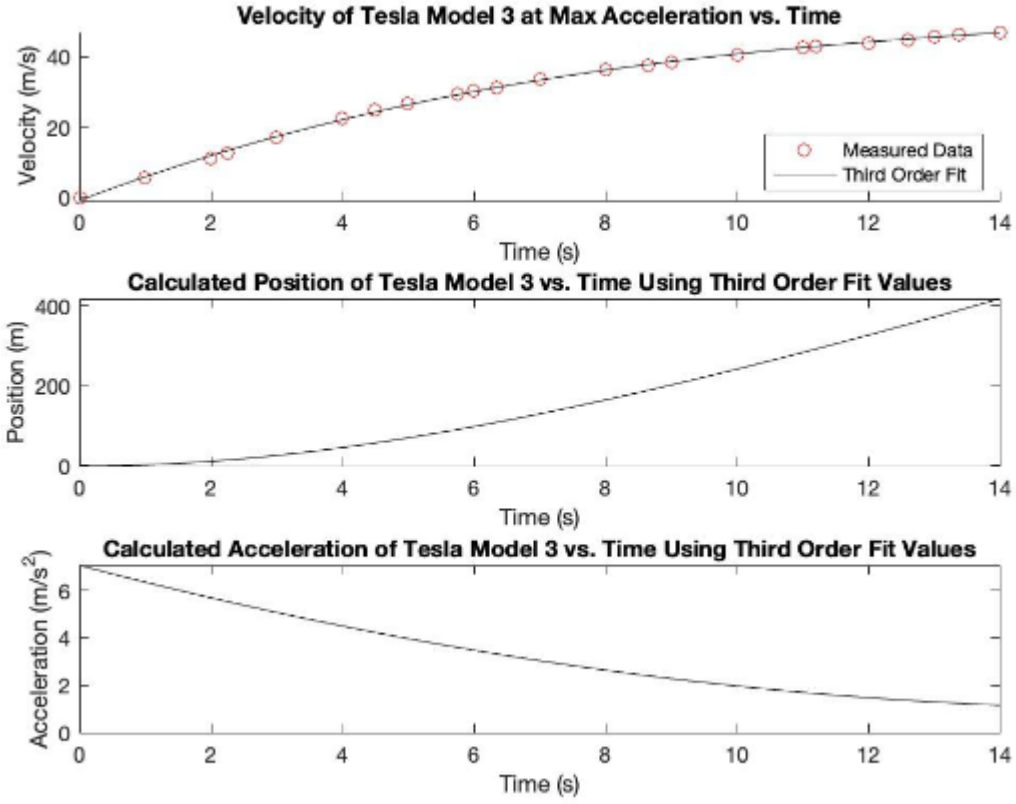
3. **Tesla Data**, created by Matthew Marinozzi

This problem considers actual data taken by third-party testers of the Tesla Model 3. The company who ran this test measured the velocity (in mph) over time (seconds) of a Tesla Model 3 at maximum acceleration. We have used different methods in the past to calculate position, velocity and acceleration, and this problem takes on a new approach to solving those three things.

The purpose of this code is to analyze and graph the velocity, acceleration and position of the car versus time. The code should achieve the following:

1. Convert the velocities from mph to m/s.
2. Create a third order fit of the velocity data.
3. Display the fit coefficients as a function of time.
4. Plot, as a subplot, velocity vs. time (from the raw data) and the velocity fit vs. time (both on the same graph), with a legend.
5. Using the coefficient values determined in step 2, calculate the position (using the idea of integration) vs. the time values used for the velocity third order fit.
6. Plot position vs. time as a subplot.
7. Repeat steps 5 and 6, this time calculating and plotting the acceleration instead of position. This would require differentiation rather than integration.

The final plot should look like this:



```
% Given (raw) data - time is in seconds, velocity is in miles per hour
time = [0 1 2 2.249 3 4 4.5 5 5.758 6 6.35 7 8 8.652 9 10 11 11.194 12 12.6 13
  13.376 14];
velocity_mph = [0 13 25 28 38 50 55.5 59.5 66 68 70 75 81 84 86 90.5 95 96.215
  98 100 102 103.499 105];

% Convert from miles per hour to meters per second (this is the correct
% conversion)
velocity_meters_per_second = velocity_mph * 0.44704;

% Determine the coefficients of a third order fit to the velocity (m/s) data
% vs.time
coeff = polyfit(time,velocity_meters_per_second,2);
fit_timeval = linspace(0,14,100);
fit_values = polyval(fit_timeval,coeff);
% Display the third order fit values as a function of t, or time
fprintf('The third order fit can be represented by the equation %.2ft^3 -
%.2ft^2 +%.2ft - %.2f, where "t" represents time.\n', coeff(1), -coeff(2),
coeff(3), -coeff(4))

% Plot velocity vs. time from the raw data, as well as the third order fit vs.
% time, on the same graph, with appropriate title and labels
figure()
subplot(3,1)
hold on
plot(time,velocity_meters_per_second,'or')
plot(fit_timeval,fit_values,'-k')
hold off
title('Velocity of Tesla Model 3 at Max Acceleration vs. Time')
xlabel('Time (s)')
ylabel('Velocity (mph)')

% Location and southeast puts the legend in the bottom right corner of the
% graph
legend('Measured Data', 'Third Order Fit', 'Location', 'southeast')

% Calculate position of the car (in meters) using the idea of integration, and
% calculate acceleration of the car (in meters per second squared) using
% derivatives. This is based on the time values used for the third order fit,
% not the original time data. It is assumed that the original position of the
% car is zero.
for i = 1:length(fit_timeval)
  position(i) = (coeff(1) / 4 * fit_timeval(i).^4) + (coeff(2) / 3 *
  fit_timeval(i).^3) + (coeff(3) / 2 * fit_timeval(i).^2) + (coeff(4) *
  fit_timeval(i));
  acceleration(i) = (coeff(1) * 3 * fit_timeval(i).^2) + (coeff(2) * 2 *
  fit_timeval(i)) + (coeff(3));
end
```

3rd order is requested

3

(coef , fit_timeval) for polyval, coef comes before x-values

I don't know why 2 and 4 are -, but it works

Subplot is (m,n,l) m by n plot at l spot should be subplot(3,1,1)

That's cool

```
% Plot position vs. time on the same figure, with appropriate title and labels
subplot(3,1,2)
plot(fit_timeval,position,'-k')
title('Calculated Position of Tesla Model 3 vs. Time Using Third Order Fit
  Values')
xlabel('Time (s)')
ylabel('Position (m)')

% Plot acceleration vs. time on the same figure, with appropriate title and
labels
subplot(3,1,3)
plot(time,acceleration,'-k')
title('Calculated Acceleration of Tesla Model 3 vs. Time Using Third Order Fit
 Values')
xlabel('Time (s)')
ylabel('Acceleration (m/s^{2})')
```

Acceleration was created on fit_timeval plot using this

4. **Sevens Out**, created by Mimi Seligman

This code simulates a simple dice game called “sevens out!”. The goal of this game is to achieve a winning score of 45 or above. A person (or matlab in this case) rolls a pair of dice, and the sum of the dice is added to their score, however if the sum is equal to seven their score resets to zero. This code includes a function to simulate the rolling of the two dice that returns their combined score. In the main script, the game is simulated, and some statistics are displayed once the game has won. Lastly, the code prints a graph that shows the players total score during each turn. There are 5 errors in the code that prevent it from functioning as intended.

```
clear;
clc;
close all
rng('shuffle');   % seed the generator

% set the winning score and current score
winning_score = 45;
current_score = 0;
num_turns = 0;

% indicates whether the game should continue (0) or has been won (1)
game_won = 0;

% simulates the game
while(game_won == 0)
  num_turns + 1;
  % calls dice roll function
  fprintf("\nRolling the dice...\n");
  pause(1);
  current_roll = roll_dice();
  if(current_roll = 7)      % resets score if 7 is rolled
    fprintf("You rolled 7! Score resets :(\n");
    current_score = 0;
  else                  % adds roll sum to score and prints out total score
    current_score = current_score + current_roll;
    fprintf("Your current score is %i.\n", current_roll);
  end

  % if winning score has been reached, ends game
  if(current_score >= winning_score)
    game_won = 1;
    fprintf("You won the game!\n");
  end

  % saves current score and number of turns in all_rolls vector
  all_rolls(1) = current_score;
  all_rolls(2) = num_turns;
end
```

```
fprintf("It took you %i turns to win the game!", num_turns);

% plots total score over all turns with a line to show the winning score
figure()
plot(all_rolls(2,:), all_rolls(1,:), "-x");
yline(45);
ylabel("Score");
xlabel("Turn Number");
xlim([1 num_turns]);
ylim([0 50]);

function score = roll_dice()
 % function to simulate rolling two dice
 % format of call: roll_dice()
 % inputs: none
 % outputs: score (sum of two dice)
 % randomly generate dice values and print them out

   die1 = randi(6);
   die2 = randi(6);
   fprintf("You rolled a %i and a %i.\n", die1, die2);

   % return the sum of the dice
   score = die1+die2;
end
```

5. **Psychic Dice**, created by Heyda Flores

While playing board games with your friends, you try to determine if you have psychic abilities. The game contains two four-sided dice, each side containing one number from 1 to 4. To figure out if you have psychic abilities, you try to guess the total sum of the numbers that will appear when you throw two dice. You will try to guess the sum, which has a minimum of 2 and a maximum of 8.

Create a code in which:

1. The computer will generate two random numbers between 1 and 4 and will then store the sum of the numbers.
2. Create a function which will determine if:

a. Your guess of the sum is correct, and the code prints you have psychic abilities

b. Your guess is wrong, and you get a second chance to guess. If you are correct, the code prints you might have psychic abilities. If you are incorrect, the code prints you are not a psychic.

c. If you enter a number less than the minimum sum or greater than the maximum of the sum, it gives a last chance to guess. If correct, the code prints you might be a psychic. If it is wrong, the code prints you are not a psychic.

```
clc;
clear;
close all;

%This is the seed generator, it guarantees a a random number each time
%MATLAB is started (this does not change).
rng('shuffle')

%Create two variables for the two dice, in which a random value from 1-4 is
%stored in each.
dice_1 = randi([1,4]);
dice_2 = randi([1,4]);

%Create the variable for the sum of the numbers stored in the two variables
%for the dice.
correct_sum = dice_1 + dice_2;

%Call the 'psychic_powers' function created below.
[p_abilities] = psychic_powers[dice_1, dice_2];

%Create 'psychic_powers' function, with output abilities, and inputs 'd_1',
%'d_2', and 'c_sum'.
function [abilities] = psychic_powers(d_1, d_2, c_sum)

%Prompt the used to guess the sum of the two numbers
user_guess = input('Enter a guess for the sum: \n');

%If the user guesses the number, print that they have psychic abilities, if
%the do not, give them a second chance. If they guess the sum on the second
%chance, print they might be psychic, since they failed on the first try.
```

```
%If they fail again, print they do not have psychic abilities.
if user_guess == c_sum
  abilities = fprintf('YOU HAVE PSYCHIC ABILITIES');
end

if user_guess ~= c_sum
  fprintf('YOU HAVE ONE GUESS LEFT\n');
  second_guess = input('Enter another guess for the sum: \n');
    if second_guess == c_sum
      abilities = fprintf('YOU MIGHT HAVE PSYCHIC ABILITIES\n');
    else second_guess ~= c_sum
      abilities = fprintf('YOU DO NOT HAVE PSYCHIC ABILITIES\n');
    end
end

%If the user enters a number that does not fit the parameters of the
%possible sums of the dice numbers, prompt them to guess again, but print
%that this will be their only chance. If they guess, print they might be
%psychic since they failed the first time. If the do not guess, print they
%are not psychic.
if user_guess < c_sum | user_guess > c_sum
  fprintf('NOT A VALID ANSWER\n');
  fprintf('GUESS AGAIN, THIS WILL BE YOUR ONLY CHANCE\n')
  guess_again = input('Enter a guess for the sum:\n');
    if user_guess == c_sum
      abilities = fprintf('YOU MIGHT HAVE PSYCHIC ABILITIES\n');
    elseif guess_again ~= c_sum
      abilities = fprintf('YOU DO NOT HAVE PSYCHIC ABILITIES\n');
     end
 end
end
```

if user_guess ~= c_sum && (user_guess > 2 && user_guess < 8)

'Else' does not have a condition

Or

Switch Places Bc 30 = sum but should be treated as out of range

Please use || not just the single one

2

8

The dice range from 2-8 for sums

guess_again

Should be in one conditional

6. **Tracking Calories**, created by Tosa Odiase

A nutritionist is looking for an efficient way to help their patients meet their calorie goals and track the amount of calories they consume. As a result, they turned to a student at Case Western Reserve to solve this problem. This Case student decided that the most efficient way to track a patient's calorie data was to create a MATLAB code. Using this code, a patient will be able to input the amount of calories and meals they have a day and receive data on the amount of calories they eat per meal and how many calories they have left to eat throughout the day. The code that this student made is meant to do the following:

- Set a calorie goal for a day
- Create a loop that
  - Asks the user how many calories they ate in a meal
  - Gives the user an option to stop inputting calories (breaking the loop)
  - Stores the total amount of calories consumed
  - Stores the total amount of meals
- Calculate the average amount of calories consumed per meal
- Calculate the amount of calories left in order to reach the daily calorie goal
- Display the amount of calories left in order to reach the daily calorie goal (descriptive with words)
- Display the average amount of calories consumed per meal (descriptive with words)

There are 5 errors in this code. Find these errors, fix them, and comment how you fixed these errors as well as why your correction is correct. Be sure to start you comment with %*** to indicate you are fixing an error.

```
% Define your daily calorie goal
total_calories = 2000;


% Initialize variables for tracking calories consumed and number of meals
eaten
calories_consumed = 5;
num_meals = 0;

% Prompt the user to enter the number of calories for each meal and update
%   variables
%Create a loop that asks for the number of calories eaten in a meal
while true
  calories = input('Enter the number of calories in your meal (or enter 0
to finish): ');

  if calories = 0
    break;                %If user inputs 0, then the if statement is broken
  end

  %Define the variable for the number of calories eaten, which the amoun
```

```
  %  of calories inputted added to the stored number of calories consumed
  calories_consumed = calories_consumed + calories;

  %Add one to the stored the number of meals, and sets that
  %  back the the variable for the number of meals
  num_meals = num_meals + 1;
end

%Define the average number of calories per meal and number of calories
%  left in the day
avg_calories_per_meal = calories_consumed / num_meals;
calories_left = total_calories - calories_consumed;

% Display the results
fprintf('You have %i calories left for the day.\n', calories_left);
fprintf('Your average meal has %i calories.\n');
```

Divide

Integer, not String

Should be floating Point %.2f

7. **Number Guessing Game**, created by Snehal Choudhury

Purpose of program: The computer "comes up" with a number from 1-100, inclusive. The user has 5 attempts to guess the computer's number, which can be a bit different from other guessing games. For each attempt, if they are too high or low, the program will let them know. If they get it right within 5 attempts, the program congratulates them and they finish the guessing game. If they don't get it right after 5 tries, the program tells them what the computer's guess was. Another unique feature of my program is that I graphed the guess # vs. the user's guess (for however many attempts they did), as well as the computer guess so the user can visually see how far off / close their guesses were from the actual number.

```
% Standard operating procedures
clear; clc; close all;

% Prevents the computer from generating 'pseudorandom' numbers
rng('shuffle')

% Computer randomly generates an integer in the range of 1-100 for the user to
% guess correctly
computer_guess = randi(1, 100);

% Boolean that stores whether the user's guess matches up with the computer's
% guess.  Seeing as we haven't started playing yet, this must be false (this
% line ISN'T wrong!)
isCorrect = false;

% Program gives the user 5 attempts to guess the computer's number correctly
% For each attempt, program tells user if their guess is too high/low or if
% they got it right (after which they exit loop)
while guess_number = 1:5

  % Using guess_number (counter variable) as vector index, this vector stores
  % the guess turn #
  num_guesses(guess_number) = guess_number;

  % Program asks user to input their guess for this particular turn
  user_guess = input("I'm thinking of a number. Give me a number in the range
   of 1-100: ", 's');

  % Using guess_number (counter variable) as vector index, this vector stores
  % the user guess for each of their attempts
  stored_guesses(guess_number) = user_guess;

  % If user guesses correctly for their given turn
  % At this point, guessing game is over
    if user_guess <= computer_guess

      % Display statement congratulating user and telling them the number of
      % tries it took
      fprintf('Congrats, you got my number right! It took you %i tries. I
```

```
       will get you next time... \n', guess_number)

      % Since user's guess correctly matches up w/ computer guess, this
      % variable must now become true
      isCorrect = true;

      % Finish the guessing game and exit the loop
      % If the user guesses correctly before all 5 attempts are used, break
      % prevents them from continuing to play
      break

  % Else, if user's guess is greater than the computer's guess
  % Display statement telling them their guess is too high
    elseif user_guess > computer_guess
      fprintf('Your guess is too high. \n')

  % Finally, if user's guess is less than the computer's guess
  % Display statement telling them their guess is too low
    elseif user_guess < computer_guess
      fprintf('Your guess is too low. \n')
    end
end

% If the user couldn't guess the computer's number in 5 tries or less
% Displays statement letting them know what the computer's number was
if isCorrect
  fprintf('Unfortunately, you were not able to guess my number in 5 tries or
    less. The number was %i. Better luck next time! \n', user_guess)
  pause(0.5) % Pause before plotting
end

% Plotting time!
% Ensures multiple graphs can be plotted in the same figure
hold on

% Plot the guess #s (x-axis) vs. the user's guesses for all of their guess
% attempts using the vectors created earlier in the loop
plot(num_guesses, stored_guesses, "*b")
yline(computer_guess, 'Width', 2) % Plot a line that shows the computer's
% guess (so the user can visually see how close/far their guesses were
% relative to the computer guess)

% Add appropriate titles, legend for the multiple plots, axis labels, and axis
% limits for the graph
title("User Guesses Relative to the Computer Guess")
legend("User Guesses", "Computer Guess")
xlabel("Guess #")
ylabel("User Guess (from 1-100)")
ylim([-10 110])
xlim([0 6])
```

8. **Prime Numbers**, created by Kailyn Smith

The code was written to teach children their multiples. This program will find the multiples of any number from 1 to 100. If it is a prime number, it will tell the user their number is prime.

```
clc;
clear;
close all;

% This asks the user for their number, creates a vector of 100 numbers from
% 1-100, and creates an empty vector where the multiples of the user's number
% will be stored.
user_num = input(Type in an integer number between 1 and 100.\n);
test_nums = linspace(1, 50, 100);
multiples = [];

% The loop tests each number from 1-100 to see if the user's number is a
% multiple of them.
for i = 1:length(test_nums)
  divide_by = test_nums(i);
  remainder = rem(user_num, divide_by);
  if remainder == 0
    multiples(end + 1) = divide_by;
end

% This will tell the user if their number is prime and what their number is
% a multiple of.
if length(multiples) == 2
  fprintf('Your number is a prime number! Its only two multiples are:\n')
  disp(multiples)
else
  fprintf('Your number is a multiple of these numbers:\n');
  disp(multiples)
end
```

9. **Training Schedules**, created by Anonymous Student

Donny is captain of the cross country team at his school and his team will be running a marathon at the end of the school year. Currently, each member of Donny's team has a different running ability. He wants to write a program that creates an individualized training schedule for each team mate. Each team member can start training whenever they want to, as long as they finish in time for the marathon.

He wrote code that takes the inputs from the user (one of Donny's team mates):
- Days until marathon
- How far the user can run now
- How many days a week the user would like to train until the marathon

Donny decided to make his program do the following:
- Display the number of days the user will be training until the marathon
- Display the change in distance from day to day
- Plot the distance the user will run for each of the training days in red circles in a graph of "distance" v.s. "training day number"

*Donny used a function so that he can re-use this code for other training purposes.

Unfortunately, Donny went for a long run right before he wrote the code and he was a little light-headed when he programmed. He could really use your help to find the mistakes he made, he knows that there are at least 5 of them. For each mistake you find, fix it, and explain what you fixed in a comment marked with %***

```
clc; clear; close all;

% Ask user how many days from tomorrow to the day of the marathon
days_til_marathon = input(How many days until you run the marathon?\n);

%Ask user how far they can comfortably run now
current_distance = input('How many miles can you currently run without over
 exerting yourself?\n');

%ask user how many days a week they want to run
training_days_per_week = input('How many days would you like to run each week?
 (enter 1-7)\n');

%set givens
marathon_distance = 26.2; %miles

%calculate total number of running days.
training_days = round(days_til_marathon * (training_days_per_week / 7));

%display number of training days
fprintf('You will run on %.0f days.\n',training_days)

%call function
[distance_increment,day,distance] =
 training_schedule(marathon_distance,current_distance,training_days);
```

input in quotes

How might I ask, does one call a non-existent function?

```
%display the distance you increase by each day
fprintf('Each day you will run %.1f miles farther than the previous day.
 \n',distance_increment)

%plot distance per day
plot(day,distance,"ok") red circles, not
ylabel('Miles')      or      black
xlabel('Day')
title('Miles per Training Day')
ylim([0 (marathon_distance)])
xticks(day) % This line of code is correct
```

10. **Squirrel Measurements**, created by Paige Greenfield

A student researcher at CWRU wants to calculate the average height and weight of the Case Quads squirrels from a set of experimental data for a health lab. The goal of the code is to use the data given and calculate the average height and weight of the squirrels, plot the graph, and then print each average to the command window.

```
%Squirrel height and weight data
height= [14, 15, 16.3, 17, 17.5, 18, 18, 18.9, 20, 22];
weight = [2.3, 1.2, 1.5, 2.8, 1.9, 1.2, 1.7, 2.4, 1, 3.0];

%Calculate the average height and weight
average_height = sum(height)/length(height);
average_weight = length(weight)/sum(weight);
average_weight = length(weight)/sum(weight);

%Print the averages
fprintf('The average height is %f\n', average_height)
fprintf('The average weight is %f\n', average_weight)

% Plot the height and weight data
figure()
plot(weight, height)
title('Height vs. Weight')
xlabel('Height (in)')
ylabel('Weight (lb)')
```

We might want to graph Weight vs. Height

11. **TIE Fighter Software Repair**, created by Jacob Gurin

A TIE fighter pilot and a group of stormtroopers have taken you hostage. They saw your Ubuntu hat, so they know you're probably enough of a nerd to be valuable. They're forcing you to repair a TIE fighter Proximity Alert System (PAS). Unfortunately for you, its software is broken, and the stormtroopers are threatening to shoot you unless you are able to fix it. They look like they have abnormally good aim, too.

The PAS functions as a sort of 360 degree radar around the craft, allowing the pilot to locate other crafts in a 2D directional ring. It plots location on a stem graph with distance in km on the y axis, with direction in degrees on the x axis (with zero being the current heading, but that isn't important). The PAS gets all of its data from one matrix with a different data type in each column, with a new row for each new craft detected.

It looks like this:
1st craft: [column 1, column 2, column 3, column 4, column 5;...
2nd craft: ...column 1, column 2, column 3, column 4, column 5]
Column 1: tracking number (starts at 1 and counts up for each craft)
Column 2: direction
Column 3: distance
Column 4: craft type (1 = friend (green), 2 = enemy (red), 3 = civilian (black))
Column 5: locking on (1 for yes, 0 for no)
It needs to do the following:
- Display where spacecraft are on a graph with different colors depending on the threat level (green is friend, red is enemy, black is civilian)
- Alert the pilot (with text in the command window) whenever an enemy craft is detected
- Alert the pilot if an enemy is locking onto them

```
clear; clc; close all

% test data (I reccomend writing it out in table form so it is easier to look
% at):
data = [1,90,50,2,1;2,270,30,1,0;3,300,70,3,0];

% examine the data for each ship
data_size = size(data(:,1));
%the line above has no errors. It outputs the demensions of the input data.
for i = 1:data_size(1)
   % alert the pilot of enemies
   if data(4,i) == 2
     input("Enemy detected\n")
   end

   % alert the pilot if the enemy is locking
   if data(i,3) == 1
     fprintf("ENEMY LOCKING-ON!\n")
   end
end

% display the crafts on a 3D, color coded, labled graph
for i =1:data_size(1)
  if data(i,4) == 1
    color = 'green';
```

organized into columns (rows,colums) should be data(i,4)

↑ Should be fprintf or disp to show output to user

colum 5 contains locking on data

for loops should be vector esque

```
  elseif data(i,4) == 2
    color = 'red';
  else
    color = 'black';
  end

stem(data(i,2),data(i,3),'color',color);
end

% make the grapg look better
xlim([0 360])
ylabel("Distance (Km)")
xlabel("Direction (degreees with 0 being forward)")
```

hold on

hold on needed to plot

12. **Equivalent Spring Constants**, created by Anonymous Student

In this problem, we are looking at three different cases of springs purely in series or parallel in a physics lab, and determining their equivalent spring constants based on individual spring constants for each spring in the system (similar to a prior homework problem). This does three different cases, as specified below:

CASE 1: Springs are in series with spring constants 1, 2, 3 and 4 N/m
CASE 2: Springs are in parallel with spring constants 20, 50, 100 and 75 N/m
CASE 3: Springs are in parallel with spring constants 0.75, 4.8, 3.2, 0.127, 0.586 , and 6.48 N/m

The code has a function that is meant to
- Find the number of springs used in the system
- Calculate a spring constant based on whether the system is in series or parallel using the following equations:

$$\text{Parallel: } k_{eq} = k_1 + k_2 + k_3 + \ldots k_n$$

$$\text{Series: } k_{eq} = \left(\frac{1}{k_1} + \frac{1}{k_2} + \frac{1}{k_3} + \ldots \frac{1}{k_n}\right)^{-1}$$

The code has a main script that should
- Provide the given values for each case
- Call the function to find the number of springs and equivalent spring constant

```
CASE 1: springs in series with spring constants 1, 2, 3, 4 N/m
equivalent spring constant: 0.480000
number of springs: 4

CASE 2: springs in parallel with spring constants of 20, 50, 100, 75 N/m
equivalent spring constant: 245.000000
number of springs: 4

CASE 3: springs in parallel, 6 spring constants with decimal values
equivalent spring constant: 15.943000
number of springs: 6
```

```
clc; clear; close all;

% find the spring constants (all the provided numbers are correct):

%case 1: springs in series with spring constants 1, 2, 3, 4 N/m
case_1 = [1 2 3 4]; %in N/m
type_1 = 'S'; %springs in series
[result_1, num_1] = k_eq(type_1, case_1);

%case 2: springs in parallel with spring constants of 20, 50, 100, 75 N/m
case_2 = [20 50 100 75]; %in N/m
type_2 = 'P'; %springs in parallel
[result_2, num_2] = k_eq(type_2, case_2);

%case 3: springs in parallel with spring constants 0.75 4.8 3.2 0.127 0.586
% 6.48 N/m
case_3 = [0.75 4.8 3.2 0.127 0.586 6.48]; %in N/m
type_3 = 'P'; %springs in parallel
[result_3, num_3] = k_eq(type_3, case_3);

%print results for all the cases
fprintf('CASE 1: springs in series with spring constants 1, 2, 3, 4 N/m')
fprintf('\nequivalent spring constant: %f \nnumber of springs: %i', result_1,
num_1)
fprintf('\n\nCASE 2: springs in parallel with spring constants of 20, 50, 100,
75 N/m')
fprintf('\nequivalent spring constant: %f \nnumber of springs: %i', num_2,
result_2)
fprintf('\n\nCASE 3: springs in parallel, 6 spring constants with decimal
values')
fprintf('\nequivalent spring constant: result_3 \nnumber of springs: num_3')

function [equiv, num] = k_eq(type, ks)
  %count the number of springs
  num = numel(ks);
  sum = 0;

  % if parallel, conduct its designated formula
  if type == 'P'
    for i = 1:length(ks)
      sum = sum + ks(i);
    end
    equiv = sum;
  end

  % if series, conduct its designated formula
  if type == 'S'
    for i = 1:length(ks)
      sum = sum + 1./ks(i);
    end
    equiv = sum;
  end

end
```

Character should be in quotes

Says Parallel but labeled 'S', in general trust comment

num_2 Number of Springs displayed Second

result_2

%f Variables Specified outside of statement %i

Series Formula shows sum^-1

Should be sum.^-1

13. **Car Repairs**, created by Bryan Sentle

In today's busy world, it's very easy to go about one's day to day and forgot things like when to repair and service their car. Therefore there is a need for something that can serve as a constant reminder for drivers to take their car for repairs, as well as estimated cost of repair. The code is supposed to take in inputs from driver (user) and using a formula, calculate when driver should take their car for a repair. The code will use a formula to calculate estimated costs.

```
% Asks user for inputs.
odo_read = input('Please input the number written on your odometer');

week10 = input('Please input an estimate of the number of journeys longer
than 10 km, that you took this week');

week40 = input('Please input an estimate of the number of journeys longer
than 40 km, that you took this week');

days = input('Please input an estimate of the number of days since your
last car service');

% Uses code to calculate and estimate number of days.
calculation = ((ode_read/1000)*((week10)^3)*(week40)*days);

% calculates cost for values that are equal to or less than 100.
if (calculation <= 100)
    cost = (calculation^3)/10000;
    proceed = 50;
    fprintf ('You can still use your car for %i days. If you go for a
repair within said number of days, cost of repair will be %i', cost,
proceed);

% calculates cost for values that are greater than 100 and less than 300.
elseif (calculation > 100 && calculation < 300)
    costa = (calculation^3)/8000;
    proceed = 30;
    fprintf ('You can still use your car for %i days. If you go for a
repair within said number of days, cost of repair will be %i', proceed,
costa);

else
    costb = (calculation^3)/5000;
    proceed = 10;
    fprintf ('You can still use your car for %i days. If you go for a
repair within said number of days, cost of repair will be %i', proceed,
costb);

end

% Displays descriptive reminder to driver.
disp ('Please always remember to keep track of your weekly travels')
```

14. **Travel Time to Airport**, created by Anonymous Student

This code was written in order to help users to determine how early they should arrive at the airport before their flight, as well as the times of day in which there is more traffic. The goal of this code is to figure out how early someone should leave for their flight based on their input on how early they'd like. The traffic was calculated from looking at the time it takes to drive from the North Residential Village to the Cleveland Airport, and then considering the hours in the day which correlated with the longest times as having the most traffic. The code should then differentiate between 'safe' times and 'risky' times based on how many hours before the flight time the user is leaving.

```
clear; clc; close all;

% Asking input for flight time and how early user would like to be
flighttime = input('What time is your flight? Please enter it in military time
and without a colon (so 11pm would be entered 2300).\n');

proximityofflighttime = input('On a scale of 1 to 3, with 3 being the earliest,
how early would you like to arrive at the airport?\n');

% For loop to calculate the time user should leave based on their preferred
% earliness; subtracting 100 is equivalent to subtracting an hour
for n = 1:3
    if proximityofflighttime = n
        leavetime = flighttime - (100*n);
    end
end

% If statement to get rid of inconsistencies where the time is negative with
% times less than 10am (i.e., if user chose 1am and subtracted 2 hours)
if leavetime <= 0 && flighttime <= 100
    leavetime = leavetime + 2400;
elseif leavetime <= 0 && flighttime < 100 && flighttime <= 200
    leavetime = leavetime + 2400;
end

% Displaying the time as a string so that it shows the zeros correctly
% (otherwise it would show up as a number having less than 4 digits)
gettime = sprintf('Considering how early you would like to be, you would leave at
%04d.');
disp(gettime);

% Creating vectors for the time and minutes of driving from dorms to airport
time = [0:1:24;0:100:2400];
traffic = [0 26 26 26 24 24 26 26 28 30 30 30 30 30 35 35 35 40 40 30 30 28 28 26
26];

% Getting a baseline for when larger amounts of traffic should be considered
averagetraffictime = 30;

% For loop to determine whether the user is getting there earlier enough
for i = 2:(numel(time(1,:))+1)

    if leavetime >= time(2,i-1) && leavetime < time(2,i)  && traffic >=
averagetraffictime
        fprintf('Be careful, there is more traffic now, and it will take %i
minutes to get to the airport from the North Residential Village.\n',traffic(i));
        if proximityofflighttime == 1
```

```
            fprintf('You should probably leave earlier, at ');
            disp(sprintf('%04d.\n',leavetime-100));
        end

    elseif leavetime >= time(2,i-1) && leavetime < time(2,i) &&
proximityofflighttime == 1 && traffic(i) < averagetraffictime
        fprintf('You are not leaving enough time to get to the airport from the
North Residential Village.\n');
    elseif leavetime >= time(2,i-1) && leavetime < time(2,i) &&
proximityofflighttime >= 2 || traffic(i) < averagetraffictime
        fprintf('You should have plenty of time to get to the airport from the
North Residential Village.\n');
    end
end
```