Challenges in Human Health:
Problem Solving and Design with Computer Science

Broad Learning Goals

1. Students will consider computer science as a potential problem solving approach to include when facing engineering design problems.
2. Students will value using computational approaches to contribute to solutions for challenges in human health.
3. Students will have increased interest in taking an introductory computer science class that includes programming.
4. Students will have increased confidence in their ability to succeed in an introductory computer science class that includes programming.

Day 2 Thursday, November 13, 2014

Specific performance goals (continued from Day 1)

6. Students will be able to write a computer program that takes a user input and uses that information to then give an output based on that input.
7. Students will be able to write a computer program that makes mathematical computations.
8. Students will be able to use computer programming to visually represent data.

Today, we are building on the introduction to computer programming and MATLAB that we started earlier this week. Three new goals related to computer science are listed immediately above. We will be accomplishing these goals in a way that demonstrates applications in human health, and in a way that takes advantage of a capability of MATLAB – making visually appealing figures of mathematical data. As in the last class, the learning is self-paced. If you complete the guided learning in this handout, you are encouraged to explore more the capabilities of MATLAB for completing computations and for making graphical representation of data. Also, if you want to deviate from this guide and experiment with these programming tools in the midst of going through this handout, you are welcome to do so.

In the last handout, you were encouraged to search online for help with learning how to use MATLAB. There is also a built-in information source in the software. In the Command Window, if you enter "help ___" where you replace the blank with some key word or function you want to learn more about, you will get documentation on that function. For example, if you type "help plot" you learn about the plot function, which we will use today.

Getting Started

Open MATLAB, and start a new .m programming file. Refer to the previous handout to remember how to do this. Save the .m file, and name it EGR100program2.m . If you like, you can choose a different name. When you run the program, you simply need to make sure

you enter the name exactly as you named the file. For example, for this program name, you would run the program by typing into the Command Window the following:
EGR100program2

Add appropriate comments to the top of the .m file to indicate the purpose of the file and to write your name on it. For example, using % signs to indicate these are comments not to be executed by the computer, you might write something like:
%This program takes user input related to human health, uses that data to make
%computations, and then visually presents the result of that computation.
%Written by One Awesome Smithie

Now, we'll start some coding, learning more tools built into MATLAB that can be used for designing solutions related to human health. We'll be using the design idea of using computer programs to help track vaccines given to a child, and to track vaccines available at a specific medical clinic, in order to help parents know when and where to take their child for a vaccine. You can learn about recommended vaccines in the US by going here: http://www.cdc.gov/vaccines/schedules/downloads/child/0-18yrs-combined-schedule-bw.pdf

Students will be able to write a computer program that takes a user input and uses that information to then give an output based on that input.
If we want to achieve the above stated goal (track vaccines for a specific child, and track vaccines available at a specific clinic), we will need to provide the computer with information about the child and about the clinic. We will ask the user (which will be you) to provide this data to get stored by the computer for later access. Alternatively, you can imagine a separate file, such as a text file or a spreadsheet, containing the data about a set of children and a set of hospitals, and you can instruct the computer program to read the data from that file.

Type these following lines of code into your EGR100program2.m file. The comments indicate what each set of code lines does. You can change the comments or the lines of code if you wish to explore more. Messing with something, including breaking it and then fixing it, is a good way to understand that thing better. This is true with computer science as much as it is with physical objects. You can always use "Save As" to save a new copy of your .m file with a new name, so you can easily go back to a previous version of your code.

```
%This section of code stores data about an individual child and the vaccines that they have
%and do not have. The data comes from the user of the program, such as a parent or a
%medical care provider.
ChildName = input('Name of child? ' ,'s');
birthdate = input('Date of birth? (MM/DD/YYYY) ','s');
disp('Did the child get the following vaccines? Enter 1 for yes, 0 for no.');
Measels = input('Measels? ');
Chicken_pox = input('Chicken pox? ');
HepB = input('Hepatitis B? ');
Tetanus = input('Tetanus? ');
```

```
%This line stores all four answers to the vaccine questions into a single
%data structure called an array.
ChildArray1 = [Measels Chicken_pox HepB Tetanus];

%Display collected data on child. The "if" statements read through and evaluate the
%previously entered data.
disp('Here is the record for the patient: ');
disp(['Name: ', ChildName]);
disp(['Date of birth (MM/DD/YYYY): ', birthdate]);
disp([ChildName, ' has the following vaccines: ']);
if ChildArray1(1) == 1
   disp('Measels');
end
if ChildArray1(2) == 1
   disp('Chicken_pox');
end
if ChildArray1(3) == 1
   disp('Hepatitis B');
end
if ChildArray1(4) == 1
   disp('Tetanus');
end
```

A few notes:
We used the array because it is an organized way to store data. Organized data is easier to then sort through later. See how later we use the positions in the array that were associated with each vaccine type? Also, we used a logic statement called an "if" statement. You can learn more about if statements by searching online or typing into the Command Window "help if" . At each if statement, we are asking if the particular location in the vaccine array for that patient indicates the patient has the vaccine. If the patient does, then we display to the screen that the patient has that vaccine. If a 0 is in the array, showing the patient does not have that vaccine, then no text gets displayed.

Students will be able to write a computer program that makes mathematical computations.

Students will be able to use computer programming to visually represent data.

Next, we will store data about a nearby clinic, adding the following lines of code to our .m file. We will also visually display the data of how many vaccine doses are available for a clinic. Again, feel free to explore, using this guide and the example code as a starting point.

```
%This section of code stores data about an individual clinic and the
%vaccines that they have and do not have available for use. The data comes
```

```
%from the user of the program, such as a medical care provider or hospital
%staff member.

ClinicName = input('Name of clinic? ' ,'s');
disp('How many doses are available for each of the following vaccines?');
MeaselsDoses = input('Measels? ');
Chicken_poxDoses = input('Chicken pox? ');
HepBDoses = input('Hepatitis B? ');
TetanusDoses = input('Tetanus? ');
ClinicArray1 = [MeaselsDoses Chicken_poxDoses HepBDoses TetanusDoses];

%Plot a bar chart of the number of doses for each vaccine.

% Create the bar chart using the bar function
figure; %This creates a new figure window.
bar(ClinicArray1); %Populate the bar chart with the data in the ClinicArray1
colormap autumn; %Specifies a color scheme. autumn is one of many color scheme options.

% Add title and axis labels
title(['Histogram of Vaccines at ', ClinicName]);
Vaccines = {'Measels', 'Ch Pox', 'HepB', 'Tetanus'}; %Stores labels for the bars.
set(gca, 'XTickLabel', Vaccines); %This is syntax for labeling the bars
ylabel('Number of doses');
```

The next section of code gives an example of using the program to making computations tracking additions and subtractions of the vaccines in a clinic. If you would like, you can choose a different scenario where computations would be part of a design solution related to tracking vaccines for a child and a clinic.

```
%Add or subtract vaccine doses, as a new shipment comes in, or as doses
%expire or are given to a patient.
%Note that we are changing the value of the previously stored data,
%overwriting the previous value. This may feel confusing at first!

disp('Enter the dose change for each vaccine. You can use positive or negative numbers.');
MeaselsDoses = MeaselsDoses + input('Measels? ');
Chicken_poxDoses = Chicken_poxDoses + input('Chicken pox? ');
HepBDoses = HepBDoses + input('Hepatitis B? ');
TetanusDoses = TetanusDoses + input('Tetanus? ');
ClinicArray2 = [MeaselsDoses Chicken_poxDoses HepBDoses TetanusDoses];

%Plot the updated bar chart of the doses available at the clinic

% Create the bar chart using the bar function
figure; %This creates a new figure window.
bar(ClinicArray2); %Populate the bar chart with the updated data in the ClinicArray2
```

```
colormap summer; %Specifies a color scheme. summer is one of many color schemes

% Add title and axis labels
title(['Updated Histogram of Vaccines at ', ClinicName]);
set(gca, 'XTickLabel', Vaccines); %This is syntax for labeling the bars
ylabel('Number of doses');
```

Designing Your Own Code

Try to write lines of code that can be used to identify a vaccine that a child does not have, and then indicate how many doses of that vaccine are available at a nearby clinic. This process would be useful to send a text message to a parent to remind them that their child is missing a vaccine and to tell them where they can get the vaccine dose. You may work individually, or work with others in the class to discuss ideas and write your code.

A Few Notes

The above example tracked one child and one clinic. What if you had thousands of children in a region served by a dozen different hospitals? The rapid repeatability of design solutions that include computer programs help substantially with larger scale problems where an action needs to be done multiple times – such as for each child and for each hospital. Think about how the example above would need to change to scale to more patients and more clinics.

Additional Exploration

Choose another problem in human health where storing and manipulating data, and/or visual representation of data is useful. For example, you might think of some need to represent data related to your team design project, or related to the Heat-a-Baby lab. Use MATLAB to input example data for the need you identify, and then for making computations with the data and creating a graphical representation of the data. If you do a Google search for "matlab graphics" you will find lot of information about options for representing data in MATLAB. The company that develops MATLAB software also has extensive information at: http://www.mathworks.com/help/pdf_doc/matlab/graphg.pdf