3 Procedure

Question 1 below, defines discrete-time signals that you will be using MATLAB to plot. The signals include discrete-time impulse (delta) and unit-step functions. To evaluate the given signals, we can define functions of our own that accept a time vector and return the discrete-time output. Here is an example of a discrete-time unit step function that can be used.

```
function y = unitstep(x)
% The unit step function, u(x).
 if (nargin ~= 1)
    disp('unit step requires 1 argument!');
    return
  end
  y = cast(x >= 0, class(x));
end
```

A small MATLAB class of simple functions has been created for this purpose, called `SimpleFunctions`. Just copy the file `SimpleFunctions.m` from the course web site and place it in the same directory as your MATLAB code. These functions can be called as in the following example.[1]

```
t = -5:5;

% Create an instance of the SimpleFunctions object.
f = SimpleFunctions();

% Evaluate the unit-step function, then plot it.
y = f.unitstep(t);

stem(t, y, 'LineWidth', 3);
```

You can now include these functions in the expressions that you create to evaluate the given functions[2] Time shifts will also work, e.g., `unitstep(t-2)`. So for example, a unit-ramp function and one shifted one time unit to the right and evaluated over the integers from -10 to 10 would be

```
t = -10:10;
unitramp = t.*unitstep(t);
unitramp_rightshift = (t-1).*unitstep(t-1);
```

1. Use MATLAB to stem plot the following discrete time signals.

[1]Make sure to read through SimpleFunctions.m so that you understand how the code works.

[2]Make sure that you do that correctly, i.e., you must use element-wise operations.

(a) $x[n] = u[n] - 2u[n-1] + u[n-4]$

(b) $x[n] = (n+2)u[n+2] - 2u[n] - nu[n-4]$

(c) $x[n] = \delta[n+1] - \delta[n] + u[n+1] - u[n-2]$

(d) $x[n] = e^{0.8n}u[n+1] + u[n]$

Include the MATLAB code and your plots in your lab report.

2. In this section MATLAB table arrays are used to process student grades obtained from a spreadsheet file. Table arrays are rectangular arrays where the row size is the same for each entry but the columns can contain different types of objects, as is often the case in spreadsheet tables. Google search "Matlab table" and you will find all kinds of examples of how to work with them.

   Download the spreadsheet `course_grades_2022.xlsx`. The file contains some student grade records[3] You can see that the student marks include 4 labs, a midterm, and 4 exam questions. Note that the entries on Line 2 give the Maximum Mark for each grade entry type. This row should obviously not be used when computing any of the questions below.

   You can use the MATLAB `readtable` function to read the xlsx file, i.e.,

```
opts = detectImportOptions('course_grades_2022.xlsx');
opts = setvartype(opts, {'ID_Number', 'Name'}, 'string');
table = readtable('course_grades_2022.xlsx', opts);
```

   This will give you a table array `table`. Note that the opts statements instruct `readtable` to import the `ID_NUMBER` and `Name` columns as strings.

   (a) Write a MATLAB function that accepts the table array from above and outputs the name and total lab mark of the person who obtained the highest total lab mark. Include the code that you wrote and a screenshot of output in your report that shows what you get when running your function.

   (b) Write a MATLAB function that accepts the table array from above and outputs the name and total exam mark of the person who obtained the highest exam mark. Include the code that you wrote and a screenshot of output in your report that shows what you get when running your function.

   (c) Write a MATLAB function that accepts the table array from above and outputs the name and final mark of the person who obtained the highest final mark. Include the code that you wrote and a screenshot of output in your report that shows what you get when calling your function.

   (d) Write MATLAB code that will add new row entries in the table for you and your lab partner. Use your actual names and MAC id numbers in the entries. You can use whatever marks you would prefer. Show the code that you wrote and a screenshot of MATLAB displaying the new table.

[3]All the data in this file, including the names, were chosen using random number/name generators.

3. Use MATLAB to do some simple image processing. From the course web site download the file:

   `ee3tp3picture2022.jpg`

   This image is a colour JPEG consisting of `1280x854` pixels. Each pixel contains 3 unsigned 8-bit integers that separately encode each RGB color in the range [0, 255]. Assume that when the image was transmitted over a network, the color channels were sent separately. When the image was reassembled, the blue channel was correctly included but the red and green channels were improperly scaled by different scaling factors. As a result, you can see that the image appears far too blue. Write a short MATLAB program that will help you fix this problem.[4]

   To get started, read the image into a matrix using the following statement.

   ```
   img = imread('ee3tp3picture2022.jpg');
   ```

   `img` will now be a `854x1280x3` matrix, i.e., `img(x, y, 1)`, `img(x, y, 2)` and `img(x, y, 3)`, gives the values of the red, green and blue channels at pixel location `(x, y)`.

   Write a short MATLAB program that will extract out each colour component (e.g., using the standard MATLAB matrix index/slice functions), rescale the red and green by different factors, then recombine the three components into a new image (i.e., using the MATLAB `cat` function to create a new image matrix). You can display a reconstructed image using the following.

   ```
   imshow(reconstructed_image);
   ```

   Do not try to automatically fix the image. Instead, use trial and error to iterate in on good scaling factors based on viewing the result.

   When you are finished include in your report: your MATLAB code with the scaling factors and a copy of the best reconstructed image that you found. You can save the final result using the following.

   ```
   imwrite(reconstructed_image, 'my_fixed_image.jpg');
   ```

4 Report

Submit a PDF report for the lab and any other files that are required. Upload files separately rather than in a zip archive. Each group (of 2 maximum) may submit a single report. To do this, your lab group must first join the same Avenue To Learn Lab 1 assignment group.

Include in your report a short description of everything that you did including all M-files and graphs.

[4]Do not use any of MATLAB's built-in image processing functions! Instead, do the improvement yourself.