Islamic University of Technology (IUT)

Organization of Islamic Cooperation (OIC)

Department of Electrical and Electronic Engineering (EEE)

INDEX:

- Group Formation with Randomly Selected Roll Numbers.
- Assignments of Exp – 1 : MATLAB Overview.
- Assignments of Exp – 2 : Shifting, Scaling & Reversal of Periodic Signal and Fourier Series.
- Programming problems (3, 6 to 15) from the book Signals & Systems Laboratory with MATLAB – by Alex Palamides & Anastasia Veloni.
- Assignments of Exp – 3 : Fourier Transform.
- Assignments of Exp – 4 : Sampling, Aliasing & Convolution.

Name	:	Md. Mohi Uddin Khan
Student ID	:	160021163
Section	:	C1
Course	:	Signals & Systems Lab
Course Code	:	EEE 4602

GUI_HOMEPAGE

```
function varargout = Homepage(varargin)
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Homepage_OpeningFcn, ...
                   'gui_OutputFcn',  @Homepage_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end

function Homepage_OpeningFcn(hObject, eventdata, handles, varargin)
% Choose default command line output for Homepage
handles.output = hObject;
% Update handles structure
guidata(hObject, handles);

function varargout = Homepage_OutputFcn(hObject, eventdata, handles)
% Get default command line output from handles structure
varargout{1} = handles.output;

% Changed here-----------------------------------
% Setting background wallpaper on UIAXES
imshow('Homepage Background Resized(1920x1080 pixel).png');

% Setting background wallpaper on Top label
top_label = imread('Homepage Background Resized for top label.png');
set(handles.home_page_label,'CData',top_label);

% Setting icon on Audio Stop Button
stop_button = imread('Audio_stop_resized(10x10).png');
set(handles.stop_sound,'CData',stop_button);
%--------------------------------------------------
% Changed Here---------------------------------
% Setting background music
cd 'E:\IUT Books\6th Semester EEE\LAB\EEE 4602_Signals & Systems\Lab 5_Home_page_for_GUI\Audio Files\WAV format'
a = randi(3);   % randomly slelecting an audio out of 3 files available
if a==1
    [audio_1 Sample_rate_1]=audioread('Beeping on CB radio_seg_1.wav');
elseif a==2
    [audio_1 Sample_rate_1]=audioread('Beeping on CB radio_seg_2.wav');
else
    [audio_1 Sample_rate_1]=audioread('Beeping on CB radio_seg_3.wav');
end
[audio_2 Sample_rate_2]=audioread('F15 Air Combat with radio chatter.wav');
audio_1 = [audio_1; audio_2]; % 2 audio files merged together
% audioplayer can play without play blocking (so your other processes continue).
global player   % declared as global; bcz, we'll later use it to pause & resume sound
player = audioplayer(audio_1,Sample_rate_1);  % for more: doc audioplayer
play(player);
% changing directory back again to mother folder
cd 'E:\IUT Books\6th Semester EEE\LAB\EEE 4602_Signals & Systems\Lab 5_Home_page_for_GUI'
%--------------------------------------------------
% Changed Here---------------------------------
% Setting background wallpaper on Index_text
```

```
index_bg = imread('Index Text Background Resized.png');
set(handles.Index_text,'CData',index_bg);

% matrix keeps each character in each byte. but {'...'} keeps total sentence in a cell. I need character by
character access, so kept in matrix form.
str1 = ['Assignment Index : '];
str2 = ['Exp-0: Group Formation With Random Rolls'];
str3 = ['Exp-1: MATLAB Overview'];
str4 = ['Exp-2: Shfting,Scaling, Signal reversal & Fourier series'];
str5 = ['Exp-3: Fourier Transform'];
str6 = ['Exp-4: Sampling, Aliasing & Convolution'];
str7 = ['Developed by: Md.Mohi Uddin Khan'];

    for i=1:numel(str1)
        str11(i) = str1(i);
        set(handles.Index_text,'String',str11);
        pause(0.05)
    end
    pause(0.3)
    for i=1:numel(str2)
        str22(i) = str2(i);
        set(handles.Index_text,'String',str22);
        pause(0.05)
    end
    pause(0.3)
    for i=1:numel(str3)
        str33(i) = str3(i);
        set(handles.Index_text,'String',str33);
        pause(0.05)
    end
    pause(0.3)
    for i=1:numel(str4)
        str44(i) = str4(i);
        set(handles.Index_text,'String',str44);
        pause(0.05)
    end
    pause(0.3)
    for i=1:numel(str5)
        str55(i) = str5(i);
        set(handles.Index_text,'String',str55);
        pause(0.05)
    end
    pause(0.3)
    for i=1:numel(str6)
        str66(i) = str6(i);
        set(handles.Index_text,'String',str66);
        pause(0.05)
    end
    pause(0.3)
    for i=1:numel(str7)
        str77(i) = str7(i);
        set(handles.Index_text,'String',str77);
        pause(0.05)
    end

    % Following things done to force user not to press any button during
    % above index_text code are being processed. Otherwise, it'll show error.
    pause(0.35)
    handles.Assignment_0.Visible = 1;
    pause(0.3)
    handles.Assignment_1.Visible = 1;
    pause(0.25)
    handles.Assignment_2.Visible = 1;
    pause(0.2)
```

```
    handles.Assignment_3_4.Visible = 1;
%-------------------------------------------------
function Assignment_0_Callback(hObject, eventdata, handles)
clc
evalin('base','clear all');  % clears the variables under current GUI. i.e: base gui

% kept delete afterwards, so that it's occured in background & user notice constant window
cd 'E:\IUT Books\6th Semester EEE\LAB\EEE 4602_Signals & Systems\Lab 0_Roll Number Generator';
Roll_Number_Generator();
delete(handles.Home_Page) % closes current GUI

function Assignment_1_Callback(hObject, eventdata, handles)
clc
evalin('base','clear all');  % clears the variables under current GUI. i.e: base gui

% kept delete afterwards, so that it's occured in background & user notice constant window
cd 'E:\IUT Books\6th Semester EEE\LAB\EEE 4602_Signals & Systems\Lab 1\Assignment Lab-1'
Asignment_Lab_1();
delete(handles.Home_Page) % closes current GUI

function Assignment_2_Callback(hObject, eventdata, handles)
clc
evalin('base','clear all');  % clears the variables under current GUI. i.e: base gui

% kept delete afterwards, so that it's occured in background & user notice constant window
cd 'E:\IUT Books\6th Semester EEE\LAB\EEE 4602_Signals & Systems\Lab 2\Assignment Lab 2'
Asignment_Lab_2();
delete(handles.Home_Page) % closes current GUI

function Assignment_3_4_Callback(hObject, eventdata, handles)
clc
evalin('base','clear all');  % clears the variables under current GUI. i.e: base gui

% kept delete afterwards, so that it's occured in background & user notice constant window
cd 'E:\IUT Books\6th Semester EEE\LAB\EEE 4602_Signals & Systems\Lab 4\Assignment Lab-4'
Assignment_Lab_4();
delete(handles.Home_Page) % closes current GUI

function stop_sound_Callback(hObject, eventdata, handles)
global player
stop(player)              % for more: doc audioplayer

function Exit_Callback(hObject, eventdata, handles)
evalin('base','clear all');  % clears the variables under current GUI. i.e: base gui
delete(handles.Home_Page) % closes current GUI
```

Output

Group Formation with Random Roll Numbers

```
function varargout = Roll_Number_Generator(varargin)
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Roll_Number_Generator_OpeningFcn, ...
                   'gui_OutputFcn',  @Roll_Number_Generator_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end

function Roll_Number_Generator_OpeningFcn(hObject, eventdata, handles, varargin)
% Choose default command line output for Roll_Number_Generator
handles.output = hObject;
% Update handles structure
guidata(hObject, handles);
% --- Outputs from this function are returned to the command line.
function varargout = Roll_Number_Generator_OutputFcn(hObject, eventdata, handles)
% Get default command line output from handles structure
varargout{1} = handles.output;

% Changed here-----------------------------------
% Setting Home button Icon on PushButton
home = imread('Home_icon_(resized 7x7).png');
set(handles.go_home,'CData',home);
%-------------------------------------------------

function Roll_From_Excel_Callback(hObject, eventdata, handles)
if get(handles.Roll_From_Excel,'Value')  % Means: If Value is 1
    msgbox('Excel File should have extension ".xlsx". Roll numbers must be kept in "Sheet 1, Column A" of
that file.',' Caution ','warn');   % For more: doc msgbox
    pause(3)  % so that user can read the msg before next code executes
end

global path Excel_Rolls    % Global variable declared
% For more: doc uigetfile
cd 'C:\';       % for security, changed to fake directory, otherwise your codes directory will be open & others will get your source code
[file path] = uigetfile({'*.xlsx'},'Select Excel File');
cd(path)        % changing directory to selected file's path

Excel_Rolls = xlsread(file,1);
Excel_Rolls = Excel_Rolls(:,1);

% Returning to code's directory after file reading
cd 'E:\IUT Books\6th Semester EEE\LAB\EEE 4602_Signals & Systems\Lab 0_Roll Number Generator'

% If too many Rolls, program will terminate for protecting PC from overload
if numel(Excel_Rolls)>20000
    msgbox('Too many Rolls causing OVERLOAD on PC. Use segment by segment Instead (e.g: 1 to 10000, then
10001 to 20000, so on..)','PC Overloaded','error');
    pause(5);
    clc;
    close all;
```

```
    Roll_Number_Generator();
end

set(handles.Starting_Roll,'string',num2str(min(Excel_Rolls(:,1))));
set(handles.Ending_Roll,'string',num2str(max(Excel_Rolls(:,1))));

function Calculate_Callback(hObject, eventdata, handles)
clc
% Input
Starting_Roll = str2num(get(handles.Starting_Roll,'string'));
Ending_Roll = str2num(get(handles.Ending_Roll,'string'));
Members_Per_Group =  str2num(get(handles.Members_Per_Group,'string'));

if isempty(Starting_Roll)
    msgbox('Empty cell detected. Enter Values Properly !!!','Invalid Input','error')   % For more: doc
msgbox
elseif isempty(Ending_Roll)
    msgbox('Empty cell detected. Enter Values Properly !!!','Invalid Input','error')   % For more: doc
msgbox
elseif isempty(Members_Per_Group)
    msgbox('Empty cell detected. Enter Values Properly !!!','Invalid Input','error')   % For more: doc
msgbox
end

% If following condition is true, program will terminate for protecting PC from overload
if Starting_Roll > Ending_Roll
    msgbox('Starting Roll can not be higher than Ending Roll.','Invalid Input','error');
    pause(5);
    clc;
    close all;
    Roll_Number_Generator();
end

% Formation of initial Roll List
global Excel_Rolls   % Global variable recalled which was created earlier
if get(handles.Roll_From_Excel,'Value') == 0
    X = Starting_Roll:Ending_Roll;
    % If too many Rolls, program will terminate for protecting PC from overload
    if numel(X)>20000
        msgbox('Too many Rolls causing OVERLOAD on PC. Use segment by segment Instead (e.g: 1 to 10000, then
10001 to 20000, so on..)','PC Overloaded','error');
        pause(5);
        clc;
        close all;
        Roll_Number_Generator();
    end
elseif get(handles.Roll_From_Excel,'Value') == 1
    for i=1:numel(Excel_Rolls(:,1))
        X(1,i) = Excel_Rolls(i,1);   % Column Matrix of Rolls Converted to Row Matrix. Bcz, whole code is
written for Row X matrix format
    end
end
% Calculation
% Odd_Even Seperation
if get(handles.Only_Odd,'Value')  % means: if Value is 1
    for i = 1:numel(X)
        X(i) = X(i)*mod(X(i),2);  % mod(3,2) = modulus, Remainder after 3/2
    end
    X = setdiff(X,0);
elseif get(handles.Only_Even,'Value')  % means: if Value is 1
    for i = 1:numel(X)
        Q(i) = X(i)*mod(X(i),2);
    end
    X = setdiff(X,Q);
```

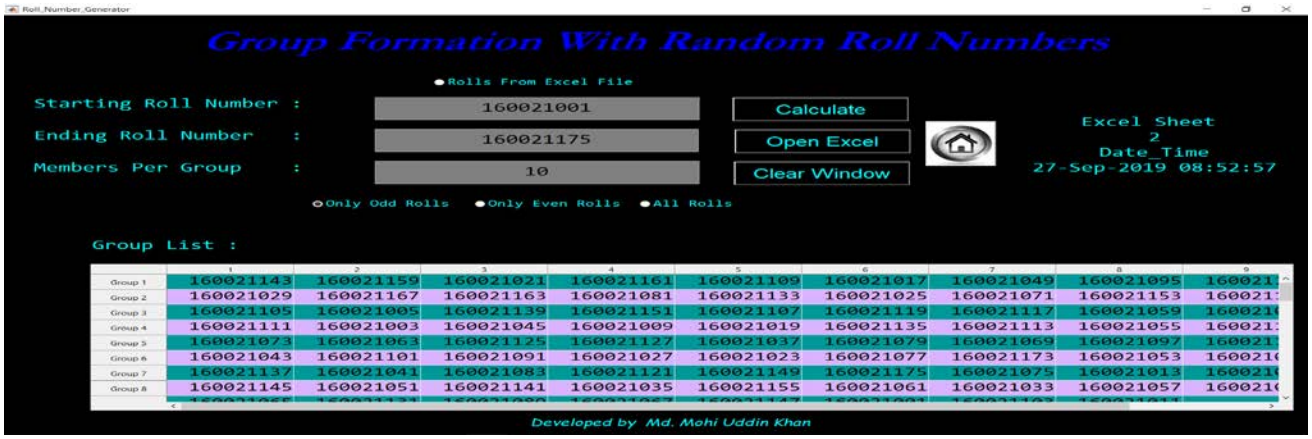
```
else
    handles.All_Roll.Value = 1;
end

% Random Rolls Stored in 'P' Matrix
for i = 1:numel(X)
    P(i) = X(randi(numel(X)));    % From 'X', choose a random integer from size of 'X'
    X = setdiff(X,P(i));          % From 'X' data, erase 'P(i)' data & update 'X' matrix
end

% Putting Random Rolls from Matrix 'P' to Matrix 'a' Groupwise
for i = 1:floor(numel(P)/Members_Per_Group)   % Block of Members_Per_Group Roll numbers declared. Round(y)
gives the nearest integer value of y.
                                        % floor(y) gives the nearest lowest integer value of y. Round isn't
                                        % used here bcz, for (Ending_Roll-Starting_Roll)>5, it takes upper
integer which causes matrix array out of size.
    a(i,:) = P(Members_Per_Group*(i-1)+1:Members_Per_Group*i);         % Extracting from P & storing in 'a'
Matrix. Members_Per_Group*(i-1)+1:Members_Per_Group*i : when i = 2, P(Members_Per_Group*(i-
1)+1:Members_Per_Group*i) = P(11:20) & so on
    P(Members_Per_Group*(i-1)+1:Members_Per_Group*i) = 0;              % Extracted positions are set 0
end
a(i+1,1:numel(P(P>0))) = P(P>0);        % Remaining values of P are extracted. P(P>0) means values of P such
that P not equals 0.

% Group Label For Excel File
for i = 1:size(a,1)
    Group(i,1) = {['Group ',num2str(i)]};
end
% Display Chart in uitable
handles.Group_List_Text.Visible = 1;
t = handles.uitable;
t.Visible = 1;
t.Data = a;

if Ending_Roll>1000000
    t.ColumnWidth = {150};
else
    t.ColumnWidth = {100};
end

global path % global variable recalled. path is the Excel File directory
if isempty(path)  % if user selects roll from file, it'll get 'path' where new Excel file of groups will be
saved. But if user gives roll inpput by himself, then he needs to select folder to save the file
    % for security, changed to fake directory, otherwise your codes directory will be open & others will get your source code
    path = uigetdir('C:\','Select Location to Save Excel File');  % for more: doc uigetdir
end
cd(path)        % changing directory to user defined path

% For more, type in command window & hit enter: doc Write Data to Excel Spreadsheets
xlswrite('Random Roll For Grouping.xlsx','Go to Next Sheets',1,'A3');
% Taking information of number of sheets in Excel file
% For more, type in command window & hit enter: doc xlsfinfo
[status,sheets,xlFormat] = xlsfinfo('Random Roll For Grouping.xlsx')
% Excel file write in new sheet
xlswrite('Random Roll For Grouping.xlsx',['Date_Time: ',string(datetime)],numel(sheets)+1,'A1')  % for more,
help datetime
% string()  function converts anything to string format. Matrix Notation
% given [] bcz it splits into 2 columns: Date_Time:     xx-xx-xxxx
xlswrite('Random Roll For Grouping.xlsx',Group',numel(sheets)+1,'A2'); % Group' = Transpose matrix of Group
xlswrite('Random Roll For Grouping.xlsx',a',numel(sheets)+1,'A3');

set(handles.Excel_Sheet_DateTime,'string',['Excel Sheet
',num2str(numel(sheets)+1),'Date_Time',string(datetime)]);
```

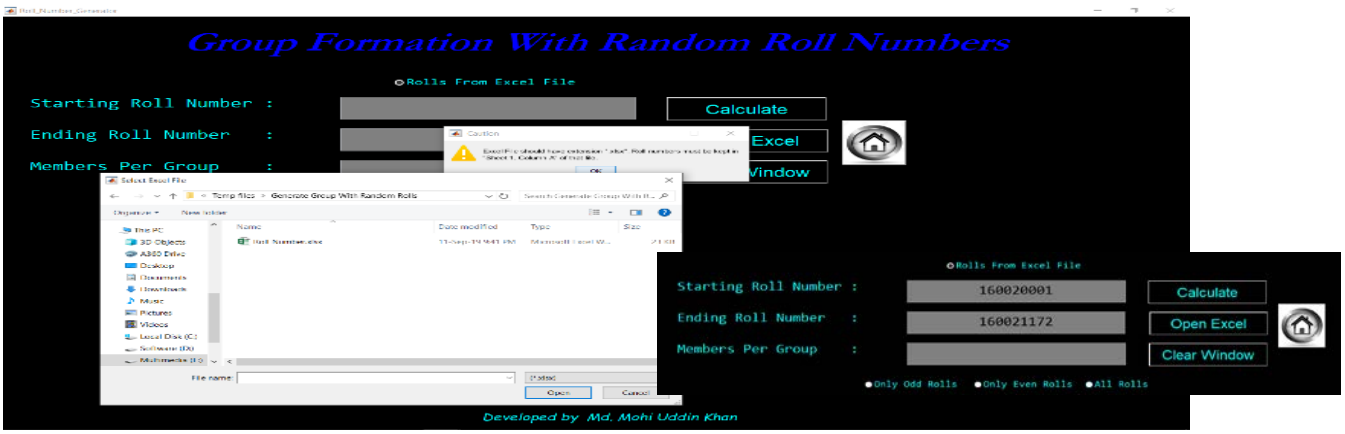
```
% Returning to code's directory after file writing
cd 'E:\IUT Books\6th Semester EEE\LAB\EEE 4602_Signals & Systems\Lab 0_Roll Number Generator'

function Open_Excel_File_Callback(hObject, eventdata, handles)
global path
% If the software is just now opened & "Open Excel" clicked, it asks user to define directory for the Exel file to open.
if isempty(path)
    cd 'C:\';     % for security, changed to fake directory, otherwise your codes directory will be open & others will get your source code
    [file path] = uigetfile({'*.xlsx'},'Select Excel File');    % For more: doc uigetfile
end

cd(path)
% For more: doc winopen
winopen('Random Roll For Grouping.xlsx');
% For more: doc msgbox
msgbox('Before Clicking on "Calculate" button again, must CLOSE this Excel File','Warning','warn');
% Returning to code's directory after file reading
cd 'E:\IUT Books\6th Semester EEE\LAB\EEE 4602_Signals & Systems\Lab 0_Roll Number Generator'

function Only_Odd_Callback(hObject, eventdata, handles)
handles.Only_Even.Value = 0;
handles.All_Roll.Value = 0;

function Only_Even_Callback(hObject, eventdata, handles)
handles.Only_Odd.Value = 0;
handles.All_Roll.Value = 0;

function All_Roll_Callback(hObject, eventdata, handles)
handles.Only_Odd.Value = 0;
handles.Only_Even.Value = 0;

function Clear_Window_Callback(hObject, eventdata, handles)
cd 'E:\IUT Books\6th Semester EEE\LAB\EEE 4602_Signals & Systems\Lab 0_Roll Number Generator'

clc
evalin('base','clear all');  % clears the variables under current GUI. i.e: base gui
Close_figures_except_GUI()

% you can write code in either format: set(handles....) & handles.xyz.... = abc
set(handles.Starting_Roll,'string',' ');
set(handles.Ending_Roll,'string',' ');
set(handles.Members_Per_Group,'string',' ');
set(handles.Excel_Sheet_DateTime,'string',' ');
handles.Only_Odd.Value = 0;
handles.Only_Even.Value = 0;
handles.All_Roll.Value = 0;
handles.Roll_From_Excel.Value = 0;
handles.Group_List_Text.Visible = 0;
handles.uitable.Visible = 0;

function go_home_Callback(hObject, eventdata, handles)
clc
evalin('base','clear all');  % clears the variables under current GUI. i.e: base gui
Close_figures_except_GUI()

delete(handles.Lab_0_GUI) % closes current GUI
cd 'E:\IUT Books\6th Semester EEE\LAB\EEE 4602_Signals & Systems\Lab 5_Home_page_for_GUI'
Homepage();
```

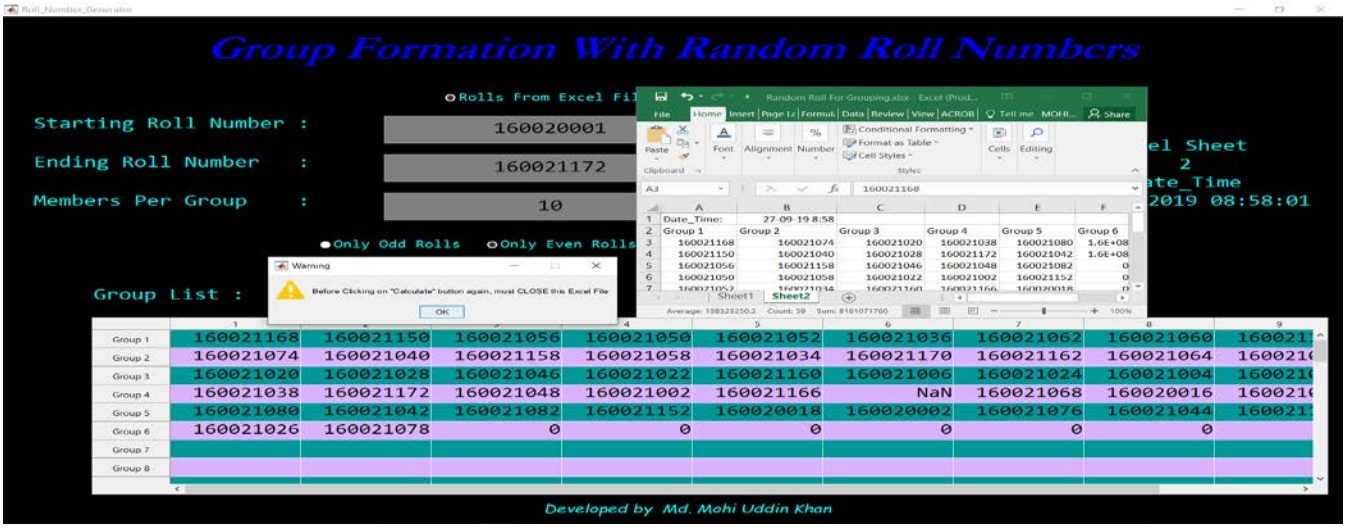
- Close_figures_except_GUI()

```
function Close_figures_except_GUI()
% Detect all figures - close the figures that are not the GUI
fh=findall(0,'type','figure');
% OR [you can use either of fh]
fh=findobj(0,'type','figure');
nfh=length(fh); % Total number of open figures, including GUI and figures with visibility 'off'
% Scan through open figures - GUI figure number is [] (i.e. size is zero)
for i = 1:nfh
    % Close all figures with a Number size is greater than zero
    if sum(size(fh(i).Number)) > 0
        figure(fh(i).Number);
        close(gcf);
    end
end
end
```

OUTPUT







Assignment of Lab – 1

- GUI Code :

```
function varargout = Asignment_Lab_1(varargin)
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Asignment_Lab_1_OpeningFcn, ...
                   'gui_OutputFcn',  @Asignment_Lab_1_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
% Initial command window clearing  ~~~~~~~~~~~~
Clc
%~~~~~~~~~~~~~~~~~~~
function Asignment_Lab_1_OpeningFcn(hObject, eventdata, handles, varargin)
handles.output = hObject;
% Update handles structure
guidata(hObject, handles);

function varargout = Asignment_Lab_1_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;

% Changed here-----------------------------------
% Setting Home button Icon on PushButton
home = imread('Home_icon_(resized 7x7).png');
set(handles.go_home,'CData',home);
%-------------------------------------------------
function one_a_Callback(hObject, eventdata, handles)
% Bunch of common tasks needs to be performed during each assignment pushbutton Callback
set(handles.input_static_text,'Visible',0);
set(handles.input,'Visible',0);
set(handles.OK,'Visible',0);
set(handles.result_table_text,'Visible',1,'string','Result Table :');
set(handles.one_a_dropdown_menu,'Visible',1,'string',{'Choose Answer...','Matrix 1','Matrix 2','Matrices
Altogether'});
set(handles.one_b_dropdown_menu,'Visible',0);
set(handles.uitable,'Visible',0);  % OR you could write 'handles.uitable.Visible = 0;' type format in all codes here
% You can also create GUI table by code which can be used just in the
% function in which the table is created. Code is given below:
% f = figure(Asignment_Lab_1);
% t = uitable(f);                 % creates new uitable in the defined figure
% t.Position = [12 40 1510 400];  % Draw a table during GUI design in design window. Click on the table &
see the position data at bottom-right corner Use that here & delete that table
                                  % for more, type in command window & hit Enter: doc uitable
% By advanced coding you can also accss this table globally.
% But drawing table manually during GUI design gives you to access the uitable
% globally; Which is used in entire code below.

function one_a_dropdown_menu_Callback(hObject, eventdata, handles)
[matrix_1, matrix_2] = Assignment_1a();
t = handles.uitable              % 'handles.uitable' gives the access to manipulate uitable drawn in GUI
design globally. It's taken as a variable t; otherwise you need to write handles.uitable.data instead of
t.data & so on
```

```
t.ColumnWidth = {'auto'};        % It's defined 'auto' here because in one_b_dropdown_menu_Callback()
function, I'll reset it to 100 pixels.
% It's rule to insert value in uitable as 'Cell data'; so {} curley braces are used.
get(t)                           % see the newly created uitable properties in command window

dropdown_menu = get(handles.one_a_dropdown_menu,'Value');   % for more: doc uidropdown

if isequal(dropdown_menu,2)
    t.Visible = 1                    % during GUI table design, I set Visibility 'off' in uitable property.
We need to make it visible now to show data.
    t.Data = matrix_1;               % OR you can write:     handles.uitable.Data = matrix_1;  OR
set(t,'Data',matrix_1);  OR   set(handles.uitable,'Data',matrix_1);
    t.ColumnName = {'Mat_1_C-1','Mat_1_C-2','Mat_1_C-3','Mat_1_C-4','Mat_1_C-5','Mat_1_C-6','Mat_1_C-
7','Mat_1_C-8','Mat_1_C-9','Mat_1_C-10'};
    t.RowName = {'Mat_1_R-1','Mat_1_R-2','Mat_1_R-3','Mat_1_R-4','Mat_1_R-5','Mat_1_R-6','Mat_1_R-
7','Mat_1_R-8','Mat_1_R-9','Mat_1_R-10'};

    get(t)                           % see how uitable properties can be changed. See the previous & new properties

elseif isequal(dropdown_menu,3)
    t.Visible = 1                    % during GUI table design, I set Visibility 'off' in uitable property.
We need to make it visible now to show data.
    t.Data = matrix_2;               % Table data changed. You can also write:     handles.uitable.Data =
matrix_1;  OR   set(t,'Data',matrix_1);  OR   set(handles.uitable,'Data',matrix_1);
    t.ColumnName = {'Mat_2_C-1','Mat_2_C-2','Mat_2_C-3','Mat_2_C-4','Mat_2_C-5','Mat_2_C-6','Mat_2_C-
7','Mat_2_C-8','Mat_2_C-9','Mat_2_C-10'};
    t.RowName = {'Mat_2_R-1','Mat_2_R-2','Mat_2_R-3','Mat_2_R-4','Mat_2_R-5','Mat_2_R-6','Mat_2_R-
7','Mat_2_R-8','Mat_2_R-9','Mat_2_R-10'};

elseif isequal(dropdown_menu,4)
    t.Visible = 1
    t.Data = matrix_1;
    p = t.Data;                      % OR you can write:    p = handles.uitable.Data;   OR
get(t,'data');    OR   get(handles.uitable,'Data');
    p(:,(end+1):(end+3)) = NaN;
    p(:,(end+1):(end+size(matrix_2,2))) = matrix_2;   % 'end' gives the position of last element.
size(matrix_2,2) gives the number of column of matrix_2
    t.Data = p;
    % OR you can write:     set(t,'data',p)    OR      other formats shown above
    t.ColumnName = {};     % Column Names Cleared
    t.RowName = {};        % Row Names Cleared

else
    t.Visible = 0;
end

function one_b_Callback(hObject, eventdata, handles)
% Bunch of common tasks needs to be performed during each assignment pushbutton Callback
set(handles.input_static_text,'Visible',0);
set(handles.input,'Visible',0);
set(handles.OK,'Visible',0);
set(handles.result_table_text,'Visible',1,'string','Result Table :');
set(handles.one_a_dropdown_menu,'Visible',0);
set(handles.one_b_dropdown_menu,'Visible',1,'string',{'Choose Answer...','5x5 Random Integer Matrix-1','3rd
Row of  Matrix-1 Replaced','Extracted 2x2 sub-matrix','Inverse of sub-matrix','Matrices Altogether'});
set(handles.uitable,'Visible',0);  % OR you could write 'handles.uitable.Visible = 0;' type format in all codes here
% Taken to global variable. Otherwise when dropdown menu is called, eachtime it calls back Assignment_1b()
which generates different random integer matrix.
global matrix_1 matrix_2 sub_matrix_2 inv_sub_matrix_2;
% Reason for writing following line of code here instead of one_b_dropdown_menu_Callback() is given in the above comment
[matrix_1, matrix_2, sub_matrix_2, inv_sub_matrix_2] = Assignment_1b();

function one_b_dropdown_menu_Callback(hObject, eventdata, handles)
t = handles.uitable              % 'handles.uitable' gives the access to manipulate uitable drawn in GUI design
```

```
globally. It's taken as a variable t; otherwise you need to write handles.uitable.data instead of t.data & so on
t.ColumnWidth =  {100};           % Set 100 pixels, so that all cells are clearly visible.
% It's rule to insert value in uitable as 'Cell data'; so {} curley braces are used.
dropdown_menu = get(handles.one_b_dropdown_menu,'Value');   % for more: doc uidropdown

global matrix_1 matrix_2 sub_matrix_2 inv_sub_matrix_2; % global variables recalled which were declared earlier

if isequal(dropdown_menu,2)
    t.Visible = 1                    % during GUI table design, I set Visibility 'off' in uitable property.
We need to make it visible now to show data.
    t.Data = matrix_1;               % OR you can write:     handles.uitable.Data = matrix_1;  OR
set(t,'Data',matrix_1);  OR   set(handles.uitable,'Data',matrix_1);
    t.ColumnName = {'C-1','C-2','C-3','C-4','C-5'};
    t.RowName = {'R-1','R-2','R-3','R-4','R-5'};

elseif isequal(dropdown_menu,3)
    t.Visible = 1                    % during GUI table design, I set Visibility 'off' in uitable property.
We need to make it visible now to show data.
    t.Data = matrix_2;              % Table data changed. You can also write:     handles.uitable.Data =
matrix_1;  OR   set(t,'Data',matrix_1);  OR   set(handles.uitable,'Data',matrix_1);
    t.ColumnName = {'C-1','C-2','C-3','C-4','C-5'};
    t.RowName = {'R-1','R-2','R-3','R-4','R-5'};

elseif isequal(dropdown_menu,4)
    t.Visible = 1
    t.Data = sub_matrix_2;
    t.ColumnName = {'C-1','C-2'};
    t.RowName = {'R-1','R-2'};

elseif isequal(dropdown_menu,5)
    t.Visible = 1
    t.Data = inv_sub_matrix_2;
    t.ColumnName = {'C-1','C-2'};
    t.RowName = {'R-1','R-2'};

elseif isequal(dropdown_menu,6)
    t.Visible = 1
    % Matrix concatenation is only possible when matrix dimensions are similar.
    % So, following things are done.
    d = zeros(10,13);
    d(:) = NaN;
    d(1:5,1:5) = matrix_1;
    d(1:5,9:13) = matrix_2;
    d(9:10,1:2) = sub_matrix_2;
    d(9:10,9:10) = inv_sub_matrix_2;
    t.Data = d;
    t.ColumnName = {};
    t.RowName = {};

else
    t.Visible = 0;
end

% --- Executes on button press in one_e.
function one_c_Callback(hObject, eventdata, handles)
% Bunch of common tasks needs to be performed during each assignment pushbutton Callback
set(handles.input_static_text,'Visible',0);
set(handles.input,'Visible',0);
set(handles.OK,'Visible',0);
set(handles.result_table_text,'Visible',0);
set(handles.one_a_dropdown_menu,'Visible',0);
set(handles.one_b_dropdown_menu,'Visible',0);
set(handles.uitable,'Visible',0);  % OR you could write 'handles.uitable.Visible = 0;' type format in all codes here
```

```
Assignment_1c();

function one_d_Callback(hObject, eventdata, handles)
% Bunch of common tasks needs to be performed during each assignment pushbutton Callback
set(handles.input_static_text,'Visible',0);
set(handles.input,'Visible',0);
set(handles.OK,'Visible',0);
set(handles.result_table_text,'Visible',0);
set(handles.one_a_dropdown_menu,'Visible',0);
set(handles.one_b_dropdown_menu,'Visible',0);
set(handles.uitable,'Visible',0);  % OR you could write 'handles.uitable.Visible = 0;' type format in all codes here
Assignment_1d()

function one_e_Callback(hObject, eventdata, handles)
% Bunch of common tasks needs to be performed during each assignment pushbutton Callback
set(handles.input_static_text,'Visible',0);
set(handles.input,'Visible',0);
set(handles.OK,'Visible',0);
set(handles.result_table_text,'Visible',0);
set(handles.one_a_dropdown_menu,'Visible',0);
set(handles.one_b_dropdown_menu,'Visible',0);
set(handles.uitable,'Visible',0);  % OR you could write 'handles.uitable.Visible = 0;' type format in all codes here
% Titles changed
set(handles.Title_bar,'string','My Student ID is ODD Number [160021163]. Try Assignment
1(d)','FontSize',30);
set(handles.one_e,'string','N/A');
pause(4);
set(handles.Title_bar,'string','Assignment of Lab - 1','FontSize',40);
set(handles.one_e,'string','1(e)');

function one_f_Callback(hObject, eventdata, handles)
% Bunch of common tasks needs to be performed during each assignment pushbutton Callback
set(handles.input_static_text,'Visible',1,'string','Enter +ve integer :','FontSize',20);
set(handles.input,'Visible',1,'string',' ');
set(handles.OK,'Visible',1,'string','OK','FontSize',20);
set(handles.result_table_text,'Visible',0);
set(handles.one_a_dropdown_menu,'Visible',0);
set(handles.one_b_dropdown_menu,'Visible',0);
set(handles.uitable,'Visible',0);  % OR you could write 'handles.uitable.Visible = 0;' type format in all codes here

function OK_Callback(hObject, eventdata, handles)

x = str2num(get(handles.input,'string'));
set(handles.input_static_text,'Visible',1,'string',['Factorial of ',num2str(x),': '],'FontSize',20);
set(handles.input,'Visible',1,'string',num2str(factorial(x)),'FontSize',20);

function one_g_Callback(hObject, eventdata, handles)
x = sin(linspace(0,10*pi,100));
y = numel(x(x>0));
% Bunch of common tasks needs to be performed during each assignment pushbutton Callback
set(handles.input_static_text,'Visible',1,'string','Number of +ve values:','FontSize',20);
set(handles.input,'Visible',1,'string',y,'FontSize',20);
set(handles.OK,'Visible',0);
set(handles.result_table_text,'Visible',0);
set(handles.one_a_dropdown_menu,'Visible',0);
set(handles.one_b_dropdown_menu,'Visible',0);
set(handles.uitable,'Visible',0);  % OR you could write 'handles.uitable.Visible = 0;' type format in all codes here

function one_h_Callback(hObject, eventdata, handles)
% Bunch of common tasks needs to be performed during each assignment pushbutton Callback
set(handles.input_static_text,'Visible',0);
set(handles.input,'Visible',0);
set(handles.OK,'Visible',0);
set(handles.result_table_text,'Visible',0);
```

```
set(handles.one_a_dropdown_menu,'Visible',0);
set(handles.one_b_dropdown_menu,'Visible',0);
set(handles.uitable,'Visible',0);  % OR you could write 'handles.uitable.Visible = 0;' type format in all codes here
Assignment_1h();

function one_i_Callback(hObject, eventdata, handles)
% Bunch of common tasks needs to be performed during each assignment pushbutton Callback
set(handles.input_static_text,'Visible',0);
set(handles.input,'Visible',0);
set(handles.OK,'Visible',0);
set(handles.result_table_text,'Visible',0);
set(handles.one_a_dropdown_menu,'Visible',0);
set(handles.one_b_dropdown_menu,'Visible',0);
set(handles.uitable,'Visible',0);  % OR you could write 'handles.uitable.Visible = 0;' type format in all codes here
Assignment_1i();

function one_j_Callback(hObject, eventdata, handles)
% Bunch of common tasks needs to be performed during each assignment pushbutton Callback
set(handles.input_static_text,'Visible',0);
set(handles.input,'Visible',0);
set(handles.OK,'Visible',0);
set(handles.result_table_text,'Visible',0);
set(handles.one_a_dropdown_menu,'Visible',0);
set(handles.one_b_dropdown_menu,'Visible',0);
set(handles.uitable,'Visible',0);  % OR you could write 'handles.uitable.Visible = 0;' type format in all codes here
Assignment_1j();

function View_Questions_Callback(hObject, eventdata, handles)
% for more type in command window 'doc figure' , 'doc Figure Properties'
f = figure('Name','Assignment-1 Questions','NumberTitle','off');
f.WindowState = 'maximized';
%f.WindowStyle = 'docked';
% Docked window can't be maximized. In normal MATLAB, docked window is
% nice to look but in executable app there is no docked window mode. So I
% also need maximized command for .exe app. So, both commands are used &
% maximized window declared first since docked window can't be maximized.
imshow('Qtn_Assignment_1.png');   % for more: doc imshow

function Reset_All_Callback(hObject, eventdata, handles)
clc
evalin('base','clear all');  % clears the variables under current GUI. i.e: base gui
Close_figures_except_GUI()
% Bunch of common tasks needs to be performed during each assignment pushbutton Callback
set(handles.input_static_text,'Visible',0);
set(handles.input,'Visible',0);
set(handles.OK,'Visible',0);
set(handles.result_table_text,'Visible',0);
set(handles.one_a_dropdown_menu,'Visible',0);
set(handles.one_b_dropdown_menu,'Visible',0);
set(handles.uitable,'Visible',0);  % OR you could write 'handles.uitable.Visible = 0;' type format in all codes here

function go_home_Callback(hObject, eventdata, handles)
clc
evalin('base','clear all');  % clears the variables under current GUI. i.e: base gui
Close_figures_except_GUI()

delete(handles.Lab_1_GUI) % closes current GUI
cd 'E:\IUT Books\6th Semester EEE\LAB\EEE 4602_Signals & Systems\Lab 5_Home_page_for_GUI'
Homepage();
```

- Code of external functions called into GUI code:

```
function [matrix_1, matrix_2] = Assignment_1a()
for i = 1:10
    if i == 1
        matrix_1(i,1) = 0;
    else
        matrix_1(i,1) = matrix_1(i-1 , 10) + 1;
    end

    for j = 2:10
        matrix_1(i,j) = matrix_1(i,j-1) + 1;
    end
end
matrix_1;
matrix_2 = matrix_1';
end
```

```
function [matrix_1, matrix_2, sub_matrix_2, inv_sub_matrix_2] = Assignment_1b()
matrix_1 = randi(5,5);
% randi(x) = random integer single value between 0 to x
% randi(x,y) = matrix of random positive integer values between 0 to min(x & y)
% rand = matrix of random positive fractional values between 0 & 1
% randn = matrix of random negative values
matrix_2 = matrix_1;
matrix_2(3,:) = rand(1,5);
matrix_2;
sub_matrix_2 = matrix_2(2:3, 4:5);
inv_sub_matrix_2 = inv(sub_matrix_2);
end
```

```
function Assignment_1c()
freq = 63; %frequency = 63 Hz
t = 0:0.00005:2; % time in seconds
omega = 2*pi*freq; % angular frequency
% Sine function using Euler's identity
sine_function = imag(exp(i*omega*t));

% for more type in command window 'doc figure' , 'doc Figure Properties'
f = figure('Name','Assignment_1c','NumberTitle','off');
f.WindowState = 'maximized';
f.WindowStyle = 'docked';
% Docked window can't be maximized. In normal MATLAB, docked window is nice to look but in executable app
% there is no docked window mode. So I also need maximized command for .exe app. So, both commands are used
% & maximized window declared first since docked window can't be maximized.
subplot(2,1,1)
plot (t, sine_function,'k');
xlabel('Time (sec)', 'FontSize',15);
ylabel('sin(2\pift)', 'FontSize',15);
title('Sine function plot where 0<t<2 second & f = 63 Hz', 'FontSize',20);

subplot(2,1,2)
plot (t, sine_function,'k','LineWidth', 2);
xlabel('Time (sec)', 'FontSize',15);
ylabel('sin(2\pift)', 'FontSize',15);
title('Sine function plot where 0<t<0.2 second & f = 63 Hz', 'FontSize',20);
xlim([0 0.2]);

hold on
grid on
% X-axis plot
```

```
x = [0.13,0.904];
y = [0.279,0.279];
annotation('arrow',x,y,'LineWidth',2,'color','k','HeadStyle','hypocycloid','HeadSize',20);

% Y-axis plot
x = [0.131,0.131];
y = [0.11,0.445];
annotation('doublearrow',x,y,'LineWidth',2,'color','k','HeadStyle','hypocycloid','HeadSize',20);
end
```

```
function Assignment_1d()
freq = 163:(163+8);         % frequency = 162 Hz
t = -1.5:0.00005:1.5;  % time in seconds

% for more type in command window 'doc figure' , 'doc Figure Properties'
f = figure('Name','Assignment_1d','NumberTitle','off');
f.WindowState = 'maximized';
f.WindowStyle = 'docked';

a = zeros(1,length(t));  % initial value
for i = 1:numel(freq)    % numel() & length() gives same output
    a = a + sin(2*pi*freq(i)*t);
    subplot(9,1,i)
    plot(t,a,'k');
    grid on;
end

pause(5);
f = figure('Name','1d [last subplot]','NumberTitle','off');
f.WindowState = 'maximized';
f.WindowStyle = 'docked';

% X-axis plot
x = [0.13,0.904];
y = [0.52,0.52];
annotation('doublearrow',x,y,'LineWidth',2,'color','k','HeadStyle','hypocycloid','HeadSize',20);

hold on;
grid on;
% Y-axis plot
x = [0.519,0.519];
y = [0.11,0.921];
annotation('doublearrow',x,y,'LineWidth',2,'color','k','HeadStyle','hypocycloid','HeadSize',20);

plot(t,a,'b');
xlabel('Time (sec)', 'FontSize',15);
ylabel('Cumulative sum of sin(2\pift)', 'FontSize',15);
title('Cumulative sum of sin(2\pift) where 163\leqf\leq163+8', 'FontSize',20); % \leq = less or equal
end
```

```
function Assignment_1h()
freq = 20; % frequency = 20 Hz
t = -0.1:0.0001:0.1; % time in seconds
% for more, type in command window & hit Enter: doc square
% x = square(omega*t,duty_cycle_in_%)
% duty cycle = (T_on/(T_on+T_off))*100% = how much % is the ON time
x = 5*square(2*pi*freq*t,60);

% for more type in command window 'doc figure' , 'doc Figure Properties'
f = figure('Name','Assignment_1h','NumberTitle','off');
f.WindowState = 'maximized';
```

```
f.WindowStyle = 'docked';

plot(t,x,'k','LineWidth',3);
xlabel('Time (sec)', 'FontSize',15);
ylabel('Amplitude (Volts)', 'FontSize',15);
title('Square wave plot. [A = 5V, f = 20Hz, D = 60%]', 'FontSize',20);
axis([-0.1 0.1 -5.6 5.6]);

hold on
grid on
% X-axis plot
x = [0.13,0.904];
y = [0.52,0.52];
annotation('doublearrow',x,y,'LineWidth',2,'color','k','HeadStyle','hypocycloid','HeadSize',20);

% Y-axis plot
x = [0.519,0.519];
y = [0.11,0.921];
annotation('doublearrow',x,y,'LineWidth',2,'color','k','HeadStyle','hypocycloid','HeadSize',20);
end
```

```
function Assignment_1i()
freq = 20; % frequency = 20 Hz
t = -0.2:0.0001:0.2; % time in seconds
% for more, type in command window & hit Enter: doc sawtooth
% x = sawtooth(omega*t)
x = 5*sawtooth(2*pi*freq*t);

% for more type in command window 'doc figure' , 'doc Figure Properties'
f = figure('Name','Assignment_1i','NumberTitle','off');
f.WindowState = 'maximized';
f.WindowStyle = 'docked';

plot(t,x,'k','LineWidth',3);
xlabel('Time (sec)', 'FontSize',15);
ylabel('Amplitude (Volts)', 'FontSize',15);
title('Sawtooth wave plot. [A = 5V, f = 20Hz]', 'FontSize',20);
axis([-0.2 0.2 -5.6 5.6]);

hold on
grid on
% X-axis plot
x = [0.13,0.904];
y = [0.52,0.52];
annotation('doublearrow',x,y,'LineWidth',2,'color','k','HeadStyle','hypocycloid','HeadSize',20);

% Y-axis plot
x = [0.519,0.519];
y = [0.11,0.921];
annotation('doublearrow',x,y,'LineWidth',2,'color','k','HeadStyle','hypocycloid','HeadSize',20);
end
```

```
function Assignment_1j()
% for more type in command window 'doc figure' , 'doc Figure Properties'
f = figure('Name','Assignment_1j','NumberTitle','off');
f.WindowState = 'maximized';
f.WindowStyle = 'docked';

% Plot 1
```

```
t = -0.5:0.01:3.5; % time in seconds
for i = 1:numel(t)
    if (t(i)<0 || t(i)>3)
        y(i) = 0;
    elseif (t(i)>=0 && t(i)<=1)
        y(i) = -t(i);
    elseif (t(i)>1 && t(i)<=2)
        y(i) = 2*t(i) - 3;
    else
        y(i) = 3 - t(i);
    end
end
subplot(2,1,1)
plot(t,y,'k','LineWidth',3);
xlabel('Time (sec)', 'FontSize',15);
ylabel('Amplitude', 'FontSize',15);
title('Triangular wave plot - a', 'FontSize',20);

hold on
grid on
% X-axis plot
x = [0.13,0.904];
y = [0.753,0.753];
annotation('doublearrow',x,y,'LineWidth',2,'color','k','HeadStyle','hypocycloid','HeadSize',20);

% Y-axis plot
x = [0.2265,0.2265];
y = [0.579,0.922];
annotation('doublearrow',x,y,'LineWidth',2,'color','k','HeadStyle','hypocycloid','HeadSize',20);

% Plot 2
t = -0.5:0.01:3.5; % time in seconds
for i = 1:numel(t)
    if (t(i)<0 || t(i)>3)
        y(i) = 0;
    elseif (t(i)>=0 && t(i)<=2)
        y(i) = t(i);
    else
        y(i) = 6 - 2*t(i);
    end
end
subplot(2,1,2)
plot(t,y,'k','LineWidth',3);
xlabel('Time (sec)', 'FontSize',15);
ylabel('Amplitude', 'FontSize',15);
title('Triangular wave plot - b', 'FontSize',20);

hold on
grid on
% X-axis plot
x = [0.13,0.904];
y = [0.111,0.1121];
annotation('doublearrow',x,y,'LineWidth',2,'color','k','HeadStyle','hypocycloid','HeadSize',20);

% Y-axis plot
x = [0.2265,0.2265];
y = [0.111,0.447];
annotation('arrow',x,y,'LineWidth',2,'color','k','HeadStyle','hypocycloid','HeadSize',20);
end
```

```
function Close_figures_except_GUI()
% Detect all figures - close the figures that are not the GUI
fh=findall(0,'type','figure');
% OR [you can use either of fh]
fh=findobj(0,'type','figure');
nfh=length(fh); % Total number of open figures, including GUI and figures with visibility 'off'
% Scan through open figures - GUI figure number is [] (i.e. size is zero)
for i = 1:nfh
    % Close all figures with a Number size is greater than zero
    if sum(size(fh(i).Number)) > 0
        figure(fh(i).Number);
        close(gcf);
    end
end
end
```

OUTPUT

- Assignment 1(a) :

Result Table : Matrix 1

	Mat_1_C-1	Mat_1_C-2	Mat_1_C-3	Mat_1_C-4	Mat_1_C-5	Mat_1_C-6	Mat_1_C-7	Mat_1_C-8	Mat_1_C-9	Mat_1_C-10
Mat_1_R-1	0	1	2	3	4	5	6	7	8	9
Mat_1_R-2	10	11	12	13	14	15	16	17	18	19
Mat_1_R-3	20	21	22	23	24	25	26	27	28	29
Mat_1_R-4	30	31	32	33	34	35	36	37	38	39
Mat_1_R-5	40	41	42	43	44	45	46	47	48	49
Mat_1_R-6	50	51	52	53	54	55	56	57	58	59
Mat_1_R-7	60	61	62	63	64	65	66	67	68	69
Mat_1_R-8	70	71	72	73	74	75	76	77	78	79
Mat_1_R-9	80	81	82	83	84	85	86	87	88	89
Mat_1_R-10	90	91	92	93	94	95	96	97	98	99

Result Table : Matrix 2

	Mat_2_C-1	Mat_2_C-2	Mat_2_C-3	Mat_2_C-4	Mat_2_C-5	Mat_2_C-6	Mat_2_C-7	Mat_2_C-8	Mat_2_C-9	Mat_2_C-10
Mat_2_R-1	0	10	20	30	40	50	60	70	80	90
Mat_2_R-2	1	11	21	31	41	51	61	71	81	91
Mat_2_R-3	2	12	22	32	42	52	62	72	82	92
Mat_2_R-4	3	13	23	33	43	53	63	73	83	93
Mat_2_R-5	4	14	24	34	44	54	64	74	84	94
Mat_2_R-6	5	15	25	35	45	55	65	75	85	95
Mat_2_R-7	6	16	26	36	46	56	66	76	86	96
Mat_2_R-8	7	17	27	37	47	57	67	77	87	97
Mat_2_R-9	8	18	28	38	48	58	68	78	88	98
Mat_2_R-10	9	19	29	39	49	59	69	79	89	99

Asignment_Lab_1

Assignment of Lab - 1

Assignment : 1(a) 1(b) 1(c) 1(d) 1(e) 1(f) 1(g) 1(h) 1(i) 1(j)

Questions

Reset All

Result Table : Matrices Altogether

0	1	2	3	4	5	6	7	8	9	NaN	NaN	NaN	0	10	20	30	40	50	60	70	80	90
10	11	12	13	14	15	16	17	18	19	NaN	NaN	NaN	1	11	21	31	41	51	61	71	81	91
20	21	22	23	24	25	26	27	28	29	NaN	NaN	NaN	2	12	22	32	42	52	62	72	82	92
30	31	32	33	34	35	36	37	38	39	NaN	NaN	NaN	3	13	23	33	43	53	63	73	83	93
40	41	42	43	44	45	46	47	48	49	NaN	NaN	NaN	4	14	24	34	44	54	64	74	84	94
50	51	52	53	54	55	56	57	58	59	NaN	NaN	NaN	5	15	25	35	45	55	65	75	85	95
60	61	62	63	64	65	66	67	68	69	NaN	NaN	NaN	6	16	26	36	46	56	66	76	86	96
70	71	72	73	74	75	76	77	78	79	NaN	NaN	NaN	7	17	27	37	47	57	67	77	87	97
80	81	82	83	84	85	86	87	88	89	NaN	NaN	NaN	8	18	28	38	48	58	68	78	88	98
90	91	92	93	94	95	96	97	98	99	NaN	NaN	NaN	9	19	29	39	49	59	69	79	89	99

Developed by: Md Mohi Uddin Khan

- Assignment 1(b) :

Result Table : 5x5 Random Integer Matrix-1

	C-1	C-2	C-3	C-4	C-5
R-1	5	1	1	1	4
R-2	5	2	5	3	1
R-3	1	3	5	5	5
R-4	5	5	3	4	5
R-5	4	5	5	5	4

Result Table : 3rd Row of Matrix-1 Replaced

	C-1	C-2	C-3	C-4	C-5
R-1	5	1	1	1	4
R-2	5	2	5	3	1
R-3	0.7577	0.7431	0.3922	0.6555	0.1712
R-4	5	5	3	4	5
R-5	4	5	5	5	4

Result Table : Extracted 2x2 sub-matrix

	C-1	C-2
R-1	3	1
R-2	0.6555	0.1712

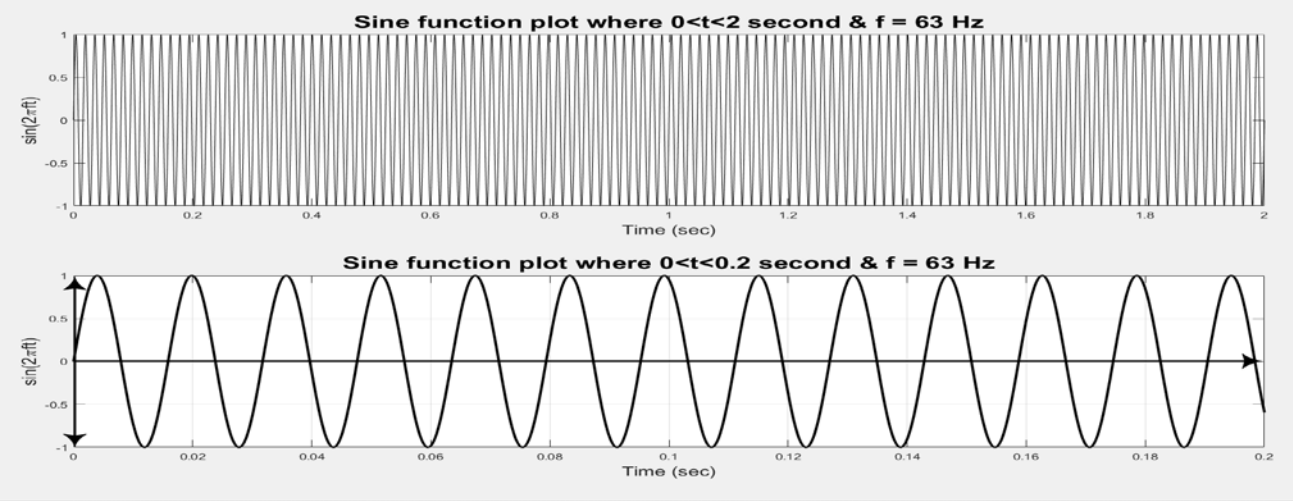
Result Table : Inverse of sub-matrix

	C-1	C-2
R-1	-1.2062	7.0463
R-2	4.6187	-21.1390

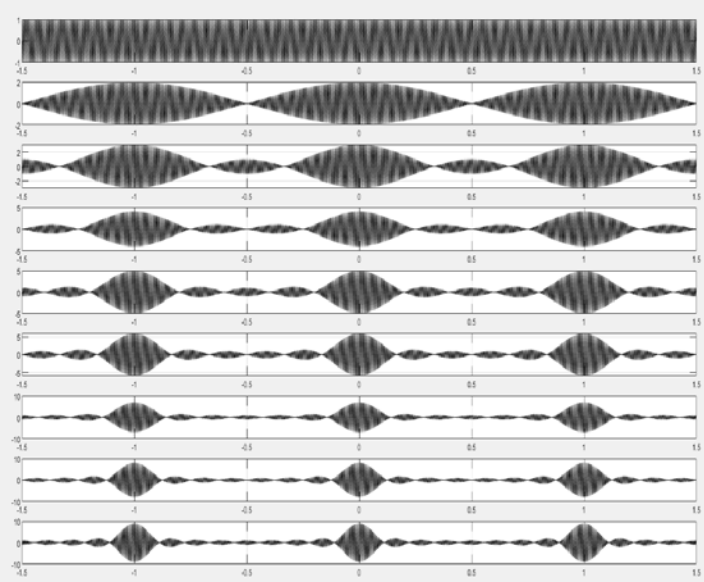
Result Table : Matrices Altogether

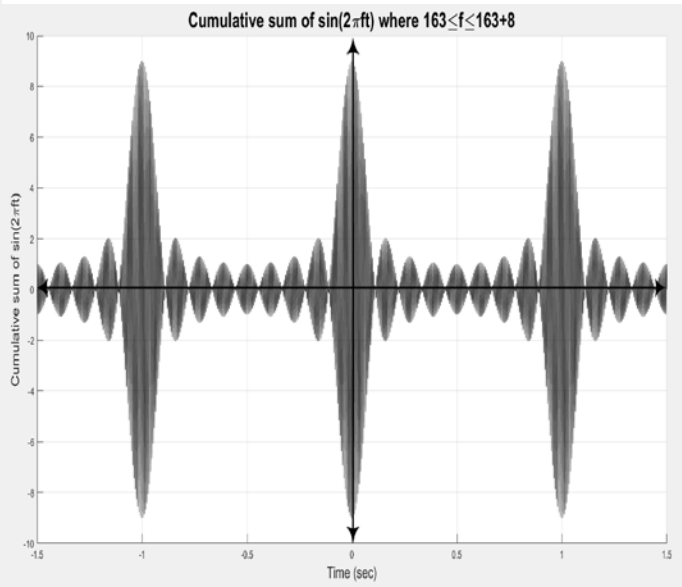
5	1	1	1	4	NaN	NaN	NaN	5	1	1	1	4
5	2	5	3	1	NaN	NaN	NaN	5	2	5	3	1
1	3	5	5	5	NaN	NaN	NaN	0.7577	0.7431	0.3922	0.6555	0.1712
5	5	3	4	5	NaN	NaN	NaN	5	5	3	4	5
4	5	5	5	4	NaN	NaN	NaN	4	5	5	5	4
NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	1	NaN	NaN	NaN	NaN	NaN	NaN	-1.2062	7.0463	NaN	NaN	NaN
0.6555	0.1712	NaN	NaN	NaN	NaN	NaN	NaN	4.6187	-21.1390	NaN	NaN	NaN

- Assignment 1(c) :

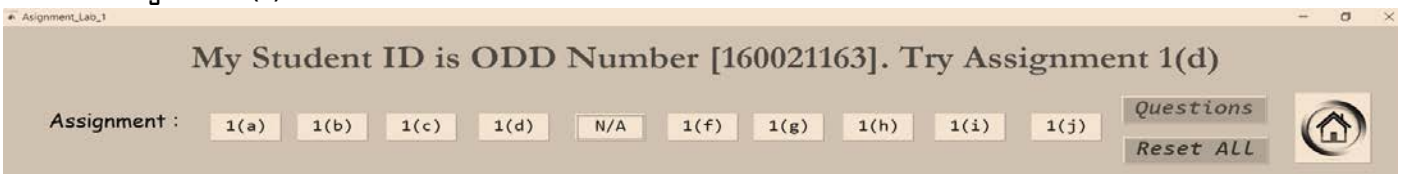


- Assignment 1(d) :

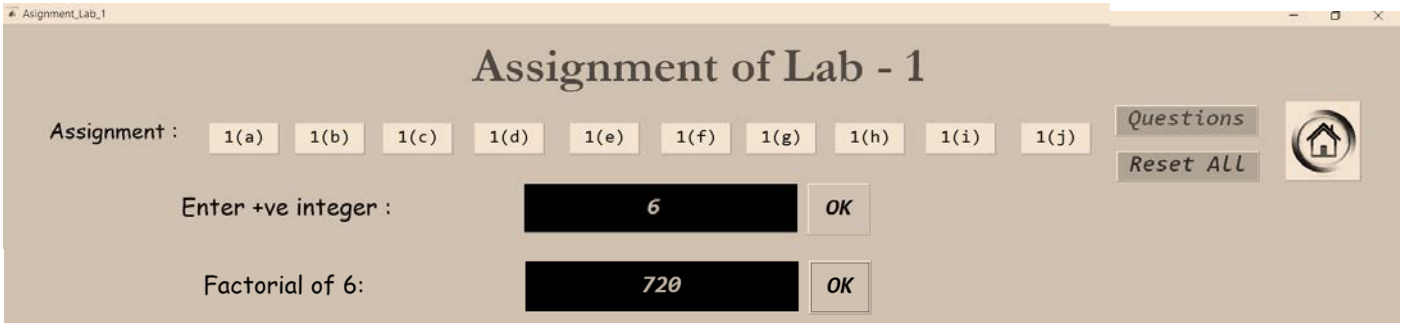




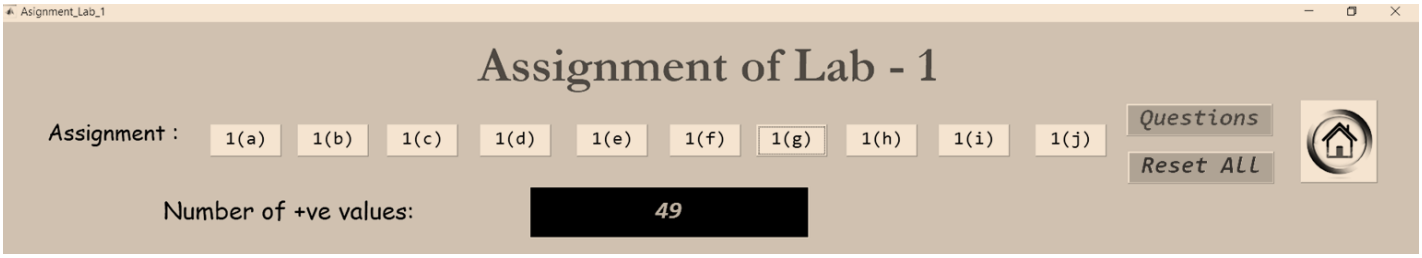
- Assignment 1(e) :



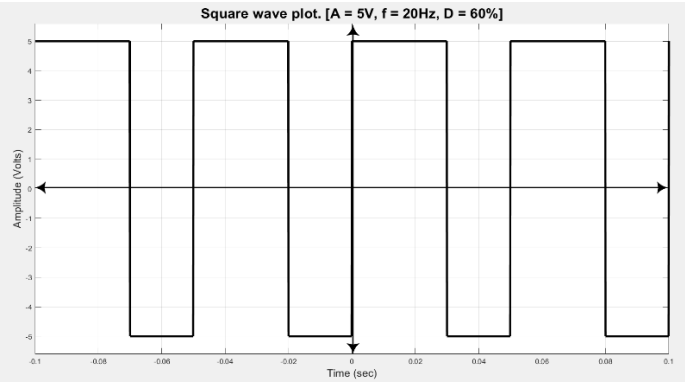
- Assignment 1(f) :



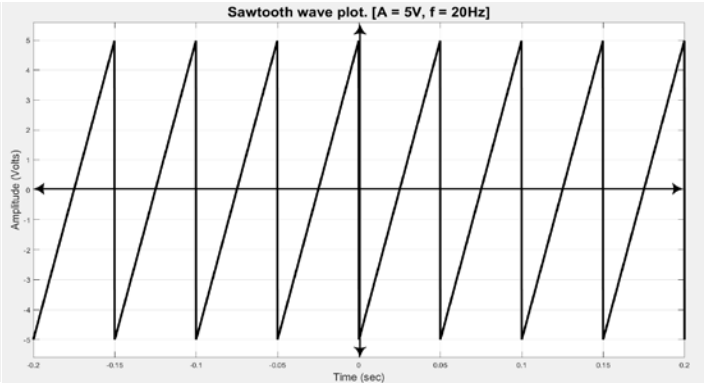
- Assignment 1(g) :



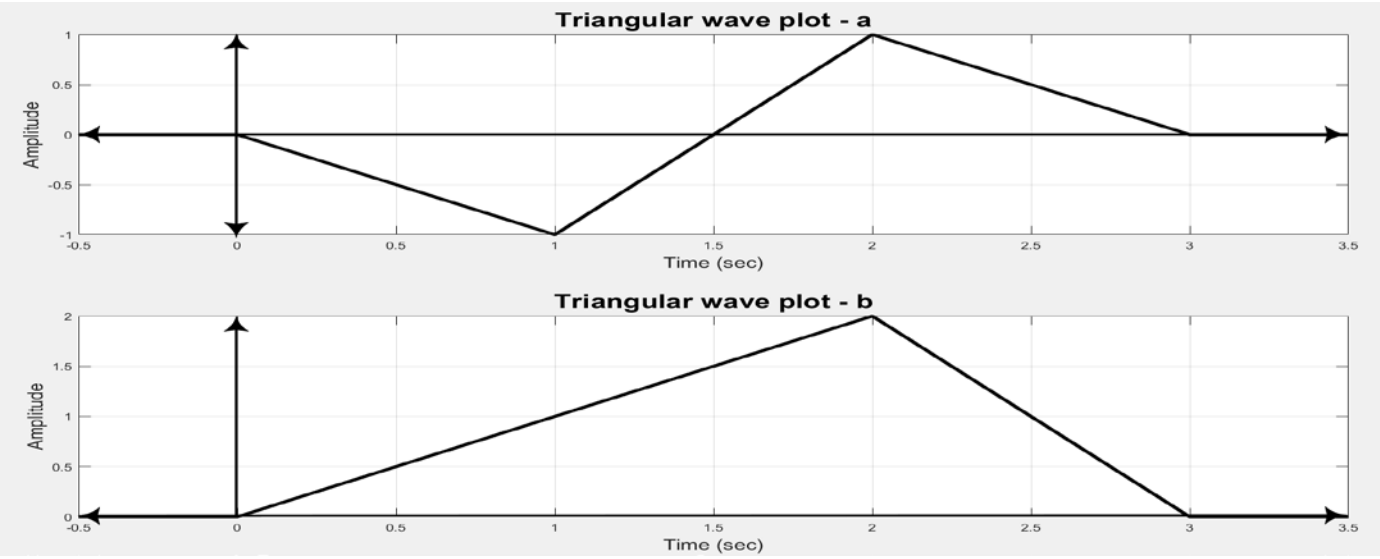
- Assignment 1(h) :



Assignment 1(i) :



- Assignment 1(j) :



ASSIGNMENT OF LAB-2 along with BOOK EXAMPLES

- GUI Code :

```
function varargout = Asignment_Lab_2(varargin)
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Asignment_Lab_2_OpeningFcn, ...
                   'gui_OutputFcn',  @Asignment_Lab_2_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
function Asignment_Lab_2_OpeningFcn(hObject, eventdata, handles, varargin)
% Choose default command line output for Asignment_Lab_2
handles.output = hObject;
% Update handles structure
guidata(hObject, handles);

function varargout = Asignment_Lab_2_OutputFcn(hObject, eventdata, handles)
% Get default command line output from handles structure
varargout{1} = handles.output;

% Changed here-----------------------------------
% Setting Home button Icon on PushButton
home = imread('Home_icon_(resized 7x7).png');
set(handles.go_home,'CData',home);
%-------------------------------------------------

function Two_a_Callback(hObject, eventdata, handles)
% to view Axes propeties, type in command window 'uiaxes'
% for more: doc uiaxes
% for more: doc uiaxes properties
if get(handles.Maximized_plot,'Value')
    Assignment_2a()
else
    % If Plot_here is selected or no radio button selected, plot will be in GUI Axes
    % Code in 'else' is the same code copied from 'if' section's called function;
    % just coded for plotting in GUI axes
    set(handles.Plot_here,'Value',1);
    % Copied from: Built_in_Square_Wave_Function()
    freq = 1; % frequency = 1 Hz
    t = -2:0.001:2; % time in seconds
    x = square(2*pi*freq*t,50);
    % changed here
    plot(handles.GUI_Graph,t,x,'k','LineWidth',3);
    set(handles.GUI_Graph,'Color',[0.3 0.75 0.93])
    %
    xlabel('Time (sec)', 'FontSize',15);
    ylabel('Amplitude', 'FontSize',15);
    title('Square wave plot. [A = 1, f = 1Hz, D = 50%]', 'FontSize',20);
    axis([-2 2 -1.1 1.1]);
    grid on
    pause(3);
    % Copied from: Assignment_2a()
```

```
    if get(handles.Maximized_plot,'Value')==0
    % Theory & code for Recontsructed square wave & Gibbs Phenomena
    web('https://gsegon.wordpress.com/2013/08/15/fourier-series-part-2-square-wave-example/');
    % for more, doc web
    pause(3);
    end

    t = linspace(-2,2,10000);   % time
    N = [1 2 3 4 7 8 50 51 100 101 1000 2000 20000 ];  % number of harmonics
    f = 0*t;                      % creates a zero valued function

    for m = 1:numel(N)
    for k=-N(m):1:N(m)
        if(k==0)                % skip the zeroth term
            continue;
        end;
        C_k = ((1)/(pi*1i*k))*(1-exp(-pi*1i*k));    % computes the k-th Fourier coefficient of the exponential form
        f_k = C_k*exp(2*pi*1i*k*t);                 % k-th term of the series
        f = f + f_k;                                % adds the k-th term to f
    end
    % Changed here
    plot(handles.GUI_Graph,t, f,'k','LineWidth', 3);
    set(handles.GUI_Graph,'Color',[0.3 0.75 0.93])
    %
    grid on;
    xlabel('Time', 'FontSize',15);
    ylabel('Function(Time)', 'FontSize',15);
    title(['Fourier synthesis of the square wave function with n = ', int2str(N(m)), '
harmonics.'],'FontSize',25);

    pause(3)
    f = 0*t;                      % re-creates a zero valued function before going for new calculation
    end
end

function Two_b_Callback(hObject, eventdata, handles)
if get(handles.Maximized_plot,'Value')
    Assignment_2b()
else
    set(handles.Plot_here,'Value',1);
    % Copied from: Assignment_2b()
    sq_wave = inline(['1*(mod(t,1)<=0.5)','-1.*(mod(t,1)>0.5 & mod(t,1)<1)'],'t');
    t = -2:0.001:2;
    % Changed here
    plot(handles.GUI_Graph,t,sq_wave(t),'k','LineWidth',3);
    set(handles.GUI_Graph,'Color',[0.3 0.75 0.93])
    %
    xlabel('Time (sec)', 'FontSize',15);
    ylabel('Amplitude', 'FontSize',15);
    title('Square wave plot. [A = 1, f = 1Hz, D = 50%]', 'FontSize',20);
    axis([-2 2 -1.2 1.2])
    grid on
end

function Two_c_Callback(hObject, eventdata, handles)
if get(handles.Maximized_plot,'Value')
    Assignment_2c()
else
    set(handles.Plot_here,'Value',1);
    % Copied From: Assignment_2c()
    triangular_wave = inline(['mod(t,1)*4.*(mod(t,1)<1/4)+','((mod(t,1)*(-
4))+2).*((mod(t,1)>=1/4)&(mod(t,1)<3/4))+','((mod(t,1)*4)-4).*((mod(t,1)>=3/4)&(mod(t,1)<1))'],'t');
    t = -6:0.0001:6;
    for i=1:2
```

```
    if i==1
        % Changed here
        plot(handles.GUI_Graph,t,triangular_wave(t),'k','LineWidth',3);  % Original plot
        set(handles.GUI_Graph,'Color',[0.3 0.75 0.93])
        %
        title('Triangular wave plot (Original).', 'FontSize',20);
    else
        % Changed here
        plot(handles.GUI_Graph,t,triangular_wave(t/3),'k','LineWidth',3);  % Just time scaling done by t/3
        set(handles.GUI_Graph,'Color',[0.3 0.75 0.93])
        %
        title('Triangular wave plot (t/3 Time Scaled).', 'FontSize',20);
    end
    xlabel('Time (sec)', 'FontSize',15);
    ylabel('Amplitude', 'FontSize',15);
    axis([0 6 -1.2 1.2])
    grid on

    if i==1
    pause(10)
    end
    end
end

function Three_Callback(hObject, eventdata, handles)
if get(handles.Maximized_plot,'Value')
    Homework_Veloni_2_8_3()
else
    set(handles.Plot_here,'Value',1);
    % Copied Form: Homework_Veloni_2_8_3()
    t = -3:0.0001:3;
    x = cos(2*pi*t) + sin(3*pi*t);
    % Changed here
    plot(handles.GUI_Graph,t,x,'k','LineWidth',3)
    set(handles.GUI_Graph,'Color',[0.3 0.75 0.93])
    %
    xlabel('Time(sec)','FontSize',20);
    ylabel('x(t)','FontSize',20);
    title('Plot of x(t) = cos(2\pit) + sin(3\pit)','FontSize',20);
    grid on
end

function Six_Callback(hObject, eventdata, handles)
if get(handles.Maximized_plot,'Value')
    Homework_Veloni_2_8_6()
else
    set(handles.Plot_here,'Value',1);
    % Copied From: Homework_Veloni_2_8_6()
    % inline function written on perunit time scale
    x = inline(['(4*mod(t,1)).*(mod(t,1)>0 & mod(t,1)<0.5) +','(4-4*mod(t,1)).*(mod(t,1)>=0.5 &
mod(t,1)<1)'],'t');
    time = 0:0.0001:4;
    t = time/4; % time scaling done to convert to original time
    axis([-4.1 8.1 0 2])
    xlabel('Time(sec)','FontSize',20);
    ylabel('y','FontSize',20);
    grid on
    % Changed here
    plot(handles.GUI_Graph,time,x(t),'k','LineWidth',3)
    set(handles.GUI_Graph,'Color',[0.3 0.75 0.93])
    %
    title('Plot of x(t)','FontSize',20);
    pause(5)
```

```
    hold on
    plot(handles.GUI_Graph,-time,x(t),'k--','LineWidth',3)
    title('Plot of x(-t)','FontSize',20);
    pause(5)

    plot(handles.GUI_Graph,2*time,x(t),'k:','LineWidth',3)
    title('Plot of x(t/2)','FontSize',20);
    pause(5)

    plot(handles.GUI_Graph,-2+time/4,x(t),'b-','LineWidth',3)
    title('Plot of x(2+4t)','FontSize',20);
    pause(5)

    plot(handles.GUI_Graph,2-time/4,x(t),'b:','LineWidth',3)
    title('Plot of x(-2-4t)','FontSize',20);

    legend({'x(t)','x(-t)','x(t/2)','x(2+4t)','x(-2-4t)'},'FontSize',18); % doc legend

    pause(5)
    title(' ');
    hold off
end

function Seven_Callback(hObject, eventdata, handles)
% For more: doc inputdlg
x = inputdlg('Shift time (t0 sec) for u(t-t0)','Enter Input', [1 50]);
if isempty(str2num(x{1}))
    % For more: doc msgbox
    msgbox('Input Value cannot be kept empty.','Error','error');
    x = inputdlg('Shift time (t0 sec) for u(t-t0)','Enter Input', [1 50]);
end
t0 = str2num(x{1});

if get(handles.Maximized_plot,'Value')
    Homework_Veloni_2_8_7(t0)
else
    set(handles.Plot_here,'Value',1);
    % Copied From: Homework_Veloni_2_8_7(t0)
    u = inline('t>=0','t');
    if t0>=0
    t = -1:0.0001:t0+4;
    else
    t = t0-4:0.0001:1;
    end
    % Changed here
    plot(handles.GUI_Graph,t,u(t-t0),'k','LineWidth',3)
    set(handles.GUI_Graph,'Color',[0.3 0.75 0.93])
    %
    xlabel('Time,t(sec)','FontSize',20);
    ylabel('u(t-t_{0})','FontSize',20);
    title(['Plot of continuous time unit step function u(t-',num2str(t0),')'],'FontSize',20);
    grid on
    axis([min(t) max(t) -0.1 1.1])
end

function Eight_Callback(hObject, eventdata, handles)
% For more: doc inputdlg
x = inputdlg('Shift time (t0 sec) for delta(t-t0)','Enter Input', [1 50]);
if isempty(str2num(x{1}))
    % For more: doc msgbox
    msgbox('Input Value cannot be kept empty.','Error','error');
    x = inputdlg('Shift time (t0 sec) for delta(t-t0)','Enter Input', [1 50]);
end
t0 = str2num(x{1});
```

```
if get(handles.Maximized_plot,'Value')
    Homework_Veloni_2_8_8(t0)
else
    set(handles.Plot_here,'Value',1);
    % Copied From: Homework_Veloni_2_8_8(t0)
    if t0>=0
    t = -1:0.0001:t0+4;
    else
    t = t0-4:0.0001:1;
    end
    impulse = t==t0;
    % Changed here
    plot(handles.GUI_Graph,t,impulse,'k','LineWidth',3)
    set(handles.GUI_Graph,'Color',[0.3 0.75 0.93])
    %
    xlabel('Time,t (sec)','FontSize',20);
    ylabel('\delta(t-t_{0})','FontSize',20);
    title(['Plot of continuous time unit impulse function \delta(t-',num2str(t0),')'],'FontSize',20);
    grid on
    ylim([-0.1 1.1])
end

function Nine_Callback(hObject, eventdata, handles)
% For more: doc inputdlg
x = inputdlg('Shift time (t0 sec) for r(t-t0)','Enter Input', [1 50]);
if isempty(str2num(x{1}))
    % For more: doc msgbox
    msgbox('Input Value cannot be kept empty.','Error','error');
    x = inputdlg('Shift time (t0 sec) for r(t-t0)','Enter Input', [1 50]);
end
t0 = str2num(x{1});

if get(handles.Maximized_plot,'Value')
    Homework_Veloni_2_8_9(t0)
else
    set(handles.Plot_here,'Value',1);
    % Copied From: Homework_Veloni_2_8_9(t0)
    r = inline('t.*(t>=0)','t');
    if t0>=0
    t = -1:0.0001:t0+4;
    else
    t = t0-4:0.0001:1;
    end
    % Changed Here
    plot(handles.GUI_Graph,t,r(t-t0),'k','LineWidth',3)
    set(handles.GUI_Graph,'Color',[0.3 0.75 0.93])
    %
    xlabel('Time,t (sec)','FontSize',20);
    ylabel('r(t-t_{0})','FontSize',20);
    title(['Plot of continuous time unit ramp function r(t-',num2str(t0),')'],'FontSize',20);
    grid on
    axis([min(t) max(t) -0.1 1.1])
end

function Ten_Callback(hObject, eventdata, handles)
% For more: doc inputdlg
x = inputdlg({'Time Period (T sec)','Shift time (t0 sec) for x(t-t0)'},'Enter Input', [1 50; 1 50]);
if isempty(str2num(x{1}))
    % For more: doc msgbox
    msgbox('Input Value cannot be kept empty.','Error','error');
    x = inputdlg({'Time Period (T sec)','Shift time (t0 sec) for x(t-t0)'},'Enter Input', [1 50; 1 50]);
elseif isempty(str2num(x{2}))
    % For more: doc msgbox
```

```
    msgbox('Input Value cannot be kept empty.','Error','error');
    x = inputdlg({'Time Period (T sec)','Shift time (t0 sec) for x(t-t0)'},'Enter Input', [1 50; 1 50]);
end
T = str2num(x{1});
t0 = str2num(x{2});

if get(handles.Maximized_plot,'Value')
    Homework_Veloni_2_8_1011(T,t0)
else
    set(handles.Plot_here,'Value',1);
    % Copied From: Homework_Veloni_2_8_1011(T,t0)
    if t0>=0
    t = 0:0.0001:t0+4;
    else
    t = t0-4:0.0001:0;
    end
    f = 1/T ; % frequency = 1/period
    sq_wave = @(t) square(2*pi*f*t,50); % doc square. doc anonymous. 50 is 50% duty cycle
    % Changed Here
    plot(handles.GUI_Graph,t,sq_wave(t-t0),'k','LineWidth',3)
    set(handles.GUI_Graph,'Color',[0.3 0.75 0.93])
    %
    xlabel('Time,t (sec)','FontSize',20);
    ylabel('Square Wave(t-t_{0})','FontSize',20);
    title(['Plot of continuous time Square Wave pT(t-',num2str(t0),')'],'FontSize',20);
    grid on
    axis([min(t) max(t) -1.1 1.1])
end

function Eleven_Callback(hObject, eventdata, handles)
Ten_Callback(hObject, eventdata, handles)

function Twelve_Callback(hObject, eventdata, handles)
% For more: doc inputdlg
x = inputdlg('Shift time (n0 sample) for u(n-n0)','Enter Input', [1 50]);
if isempty(str2num(x{1}))
    % For more: doc msgbox
    msgbox('Input Value cannot be kept empty.','Error','error');
    x = inputdlg('Shift time (n0 sample) for u(n-n0)','Enter Input', [1 50]);
end
n0 = str2num(x{1});

if get(handles.Maximized_plot,'Value')
    Homework_Veloni_2_8_12(n0)
else
    set(handles.Plot_here,'Value',1);
    % Copied From: Homework_Veloni_2_8_12(n0)
    u = inline('n>=0','n');
    n0 = round(n0);         % bcz, discrete function is valid only at integer time (e.g: at 5, not at 5.6)
    if n0>=0
    n = -1:1:n0+4;   % bcz, discrete function is valid only at integer time (e.g: at 5, not at 5.6)
    else
    n = n0-4:1:1;
    end
    % Changed Here
    stem(handles.GUI_Graph,n,u(n-n0),'k','filled','LineWidth',3,'MarkerSize',10)
    set(handles.GUI_Graph,'Color',[0.3 0.75 0.93])
    %
    xlabel('Time,n (sec)','FontSize',20);
    ylabel('u(n-n_{0})','FontSize',20);
    title(['Plot of discrete time unit step function u(n-',num2str(n0),')'],'FontSize',20);
    grid on
    axis([min(n) max(n) -0.1 1.1])
```

```
end

function Thirteen_Callback(hObject, eventdata, handles)
% For more: doc inputdlg
x = inputdlg('Shift time (n0 sample) for delta(n-n0)','Enter Input', [1 50]);
if isempty(str2num(x{1}))
    % For more: doc msgbox
    msgbox('Input Value cannot be kept empty.','Error','error');
    x = inputdlg('Shift time (n0 sample) for delta(n-n0)','Enter Input', [1 50]);
end
n0 = str2num(x{1});

if get(handles.Maximized_plot,'Value')
    Homework_Veloni_2_8_13(n0)
else
    set(handles.Plot_here,'Value',1);
    % Copied From: Homework_Veloni_2_8_13(n0)
    n0 = round(n0);       % bcz, discrete function is valid only at integer time (e.g: at 5, not at 5.6)
    if n0>=0
    n = -1:1:n0+4;
    else
    n = n0-4:1:1;
    end
    impulse = n==n0;
    % Changed Here
    stem(handles.GUI_Graph,n,impulse,'k','filled','LineWidth',3,'MarkerSize',10,'Marker','^')
    set(handles.GUI_Graph,'Color',[0.3 0.75 0.93])
    %
    xlabel('Time,n (sec)','FontSize',20);
    ylabel('\delta(n-n_{0})','FontSize',20);
    title(['Plot of discrete time unit impulse function \delta(n-',num2str(n0),')'],'FontSize',20);
    grid on
    ylim([-0.1 1.1])
end

function Fourteen_Callback(hObject, eventdata, handles)
% For more: doc inputdlg
str = {'Time Limit (n samples)','Shift time (n0 sample) for x(n-n0)','Signal Function (e.g: cos(4*pi*n/11) +
2.*i*sin(6*pi*n/11)  OR  2*sin(14*pi*n/19) + cos(10*pi*n/19) + 1  etc.)'};
x = inputdlg(str,'Enter Input', [1 50; 1 50; 1 100]);

if isempty(str2num(x{1}))
    % For more: doc msgbox
    msgbox('Input Value cannot be kept empty.','Error','error');
    x = inputdlg(str,'Enter Input', [1 50; 1 50; 1 100]);
elseif isempty(str2num(x{2}))
    % For more: doc msgbox
    msgbox('Input Value cannot be kept empty.','Error','error');
    x = inputdlg(str,'Enter Input', [1 50; 1 50; 1 100]);
elseif isempty(x{3})
    % For more: doc msgbox
    msgbox('Input Value cannot be kept empty.','Error','error');
    x = inputdlg(str,'Enter Input', [1 50; 1 50; 1 100]);
end
n = str2num(x{1});
n0 = str2num(x{2});
signal_fn = x{3};

if get(handles.Maximized_plot,'Value')
    Homework_Veloni_2_8_14(signal_fn,n,n0)
else
    set(handles.Plot_here,'Value',1);
    % Copied From: Homework_Veloni_2_8_14(signal_fn,n,n0)
    n0 = round(n0);       % bcz, discrete function is valid only at integer time (e.g: at 5, not at 5.6)
```

```
    n = -round(n):1:round(n);
    if abs(n0)>max(abs(n))
    msgbox('Shifted time exceeds Time Limit given.','Invalid input','error');
    return    % breaks & terminates the function. 'break' is used inside 'Loop' & 'return' is used inside
'if-else'
    end
    signal = inline(signal_fn,'n')
    % Changed Here
    stem(handles.GUI_Graph,n,signal(n-n0),'k','filled','LineWidth',3,'MarkerSize',10)
    set(handles.GUI_Graph,'Color',[0.3 0.75 0.93])
    %
    xlabel('Time,n (sec)','FontSize',20);
    ylabel('x(n-n_{0})','FontSize',20);
    title(['Plot of time shifted (n-',num2str(n0),') discrete time input function ',signal_fn
],'FontSize',20);
    grid on
end

function Fifteen_Callback(hObject, eventdata, handles)
% For more: doc inputdlg
str = {'Time Limit (n samples)','Time Scaling Co-efficient (a) for x(a.n)','Signal Function (e.g:
cos(4*pi*n/11) + 2.*i*sin(6*pi*n/11)  OR  2*sin(14*pi*n/19) + cos(10*pi*n/19) + 1  etc.)'};
x = inputdlg(str,'Enter Input', [1 50; 1 50; 1 100]);

if isempty(str2num(x{1}))
    % For more: doc msgbox
    msgbox('Input Value cannot be kept empty.','Error','error');
    x = inputdlg(str,'Enter Input', [1 50; 1 50; 1 100]);
elseif isempty(str2num(x{2}))
    % For more: doc msgbox
    msgbox('Input Value cannot be kept empty.','Error','error');
    x = inputdlg(str,'Enter Input', [1 50; 1 50; 1 100]);
elseif isempty(x{3})
    % For more: doc msgbox
    msgbox('Input Value cannot be kept empty.','Error','error');
    x = inputdlg(str,'Enter Input', [1 50; 1 50; 1 100]);
end
n = str2num(x{1});
a = str2num(x{2});
signal_fn = x{3};

if get(handles.Maximized_plot,'Value')
    Homework_Veloni_2_8_15(signal_fn,n,a)
else
    set(handles.Plot_here,'Value',1);
    % Copied From: Homework_Veloni_2_8_15(signal_fn,n,a)
    n = -round(n):1:round(n);
    %% time scaling operation
    m = n;
    n1 = 0;
    j = 1;
    for i = 1:numel(n)
        if mod((n(1,i)/a),1)==0
            n1(1,j) = n(1,i)/a;   % bcz, x(a*n) divides time axis by a. i.e: n/a
            j = j+1;
        else
             m(1,i) = 0;     % the values of n I don't need are set to 0
        end
    end
    m
    i = find(n==0)
    m1 = setdiff(m(1,1:i-1),0)   % for more, doc setdiff
    m2 = setdiff(m(1,i+1:numel(n)),0)
    m  = [m1 0 m2]
```

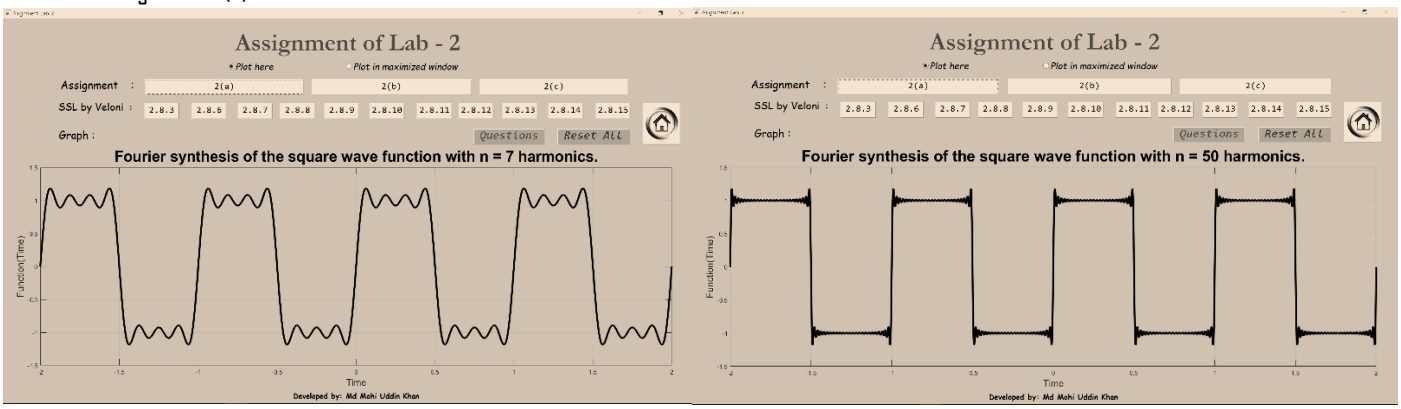
```
    n1
% ------------------------
    signal = inline(signal_fn,'n')
    % Changed Here
    stem(handles.GUI_Graph,n,signal(n),'k','filled','LineWidth',3,'MarkerSize',10)
    set(handles.GUI_Graph,'Color',[0.3 0.75 0.93])
    %
    xlabel('Time,n (sec)','FontSize',20);
    ylabel('x(n)','FontSize',20);
    title(['Plot of original discrete time input function ',signal_fn ],'FontSize',20);
    grid on
    pause(5)
    % Changed Here
    stem(handles.GUI_Graph,n1,signal(m),'k','filled','LineWidth',3,'MarkerSize',10)
    set(handles.GUI_Graph,'Color',[0.3 0.75 0.93])
    %
    xlabel('Time,n (sec)','FontSize',20);
    ylabel('x(a*n)','FontSize',20);
    title(['Plot of time scaled (',num2str(a),'n) discrete time input function ',signal_fn ],'FontSize',20);
end

function Plot_here_Callback(hObject, eventdata, handles)
set(handles.Maximized_plot,'Value',0);

function Maximized_plot_Callback(hObject, eventdata, handles)
set(handles.Plot_here,'Value',0);

function View_Questions_Callback(hObject, eventdata, handles)
% for more type in command window 'doc figure' , 'doc Figure Properties'
f = figure('Name','Assignment-2 Questions','NumberTitle','off');
f.WindowState = 'maximized';
%f.WindowStyle = 'docked';
% Docked window can't be maximized. In normal MATLAB, docked window is
% nice to look but in executable app there is no docked window mode. So I
% also need maximized command for .exe app. So, both commands are used &
% maximized window declared first since docked window can't be maximized.
imshow('Qtn_Assignment_2.png');   % for more: doc imshow

f = figure('Name','Assignment-2 Book Questions','NumberTitle','off');
f.WindowState = 'maximized';
f.WindowStyle = 'docked';
imshow('Qtn_Assignment_2_book.png');   % for more: doc imshow

function Reset_All_Callback(hObject, eventdata, handles)
clc
evalin('base','clear all');  % clears the variables under current GUI. i.e: base gui
Close_figures_except_GUI()
% Clearing UIAXES
plot(handles.GUI_Graph,0,0)
set(handles.GUI_Graph,'Color',[0.3 0.75 0.93])
grid on
%
set(handles.Plot_here,'Value',0);
set(handles.Maximized_plot,'Value',0);

function go_home_Callback(hObject, eventdata, handles)
clc
evalin('base','clear all');  % clears the variables under current GUI. i.e: base gui
Close_figures_except_GUI()

delete(handles.Lab_2_GUI) % closes current GUI
cd 'E:\IUT Books\6th Semester EEE\LAB\EEE 4602_Signals & Systems\Lab 5_Home_page_for_GUI'
Homepage();
```

- Code of external functions called into GUI code:

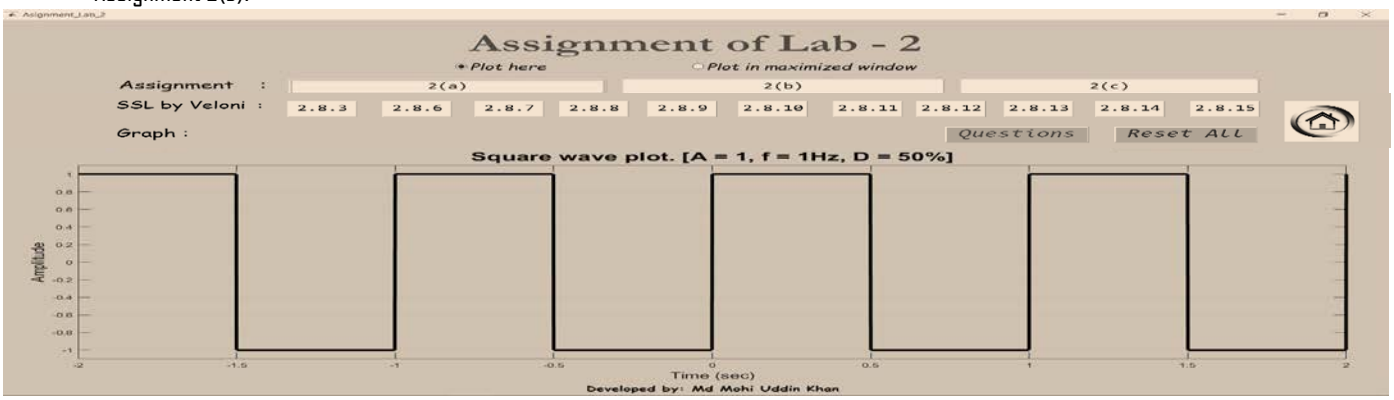
```
function Close_figures_except_GUI()
% Detect all figures - close the figures that are not the GUI
fh=findall(0,'type','figure');
% OR [you can use either of fh]
fh=findobj(0,'type','figure');
nfh=length(fh); % Total number of open figures, including GUI and figures with visibility 'off'
% Scan through open figures - GUI figure number is [] (i.e. size is zero)
for i = 1:nfh
    % Close all figures with a Number size is greater than zero
    if sum(size(fh(i).Number)) > 0
        figure(fh(i).Number);
        close(gcf);
    end
end
end
```

Output Graphs

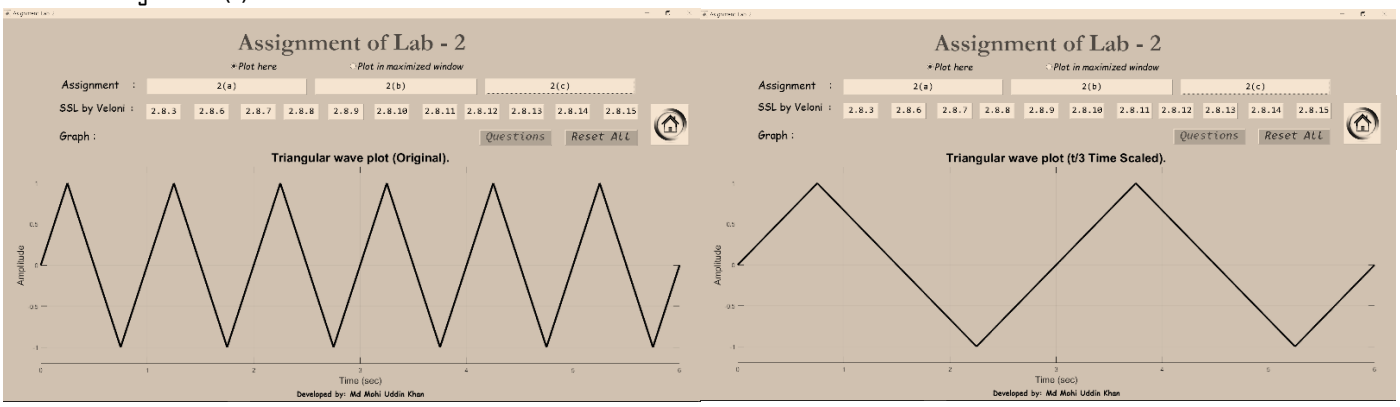
- Assignment 2(a):



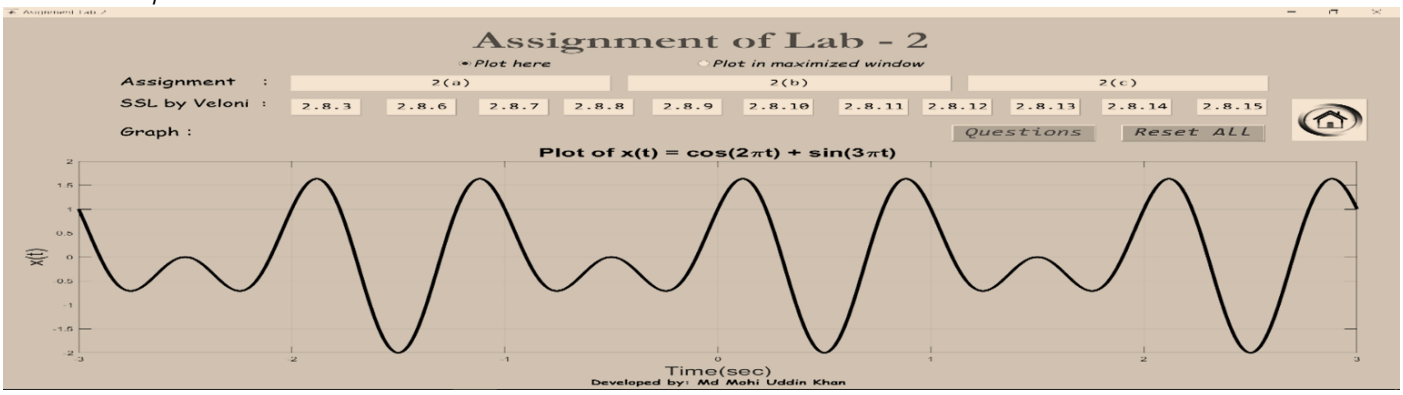
- Assignment 2(b):



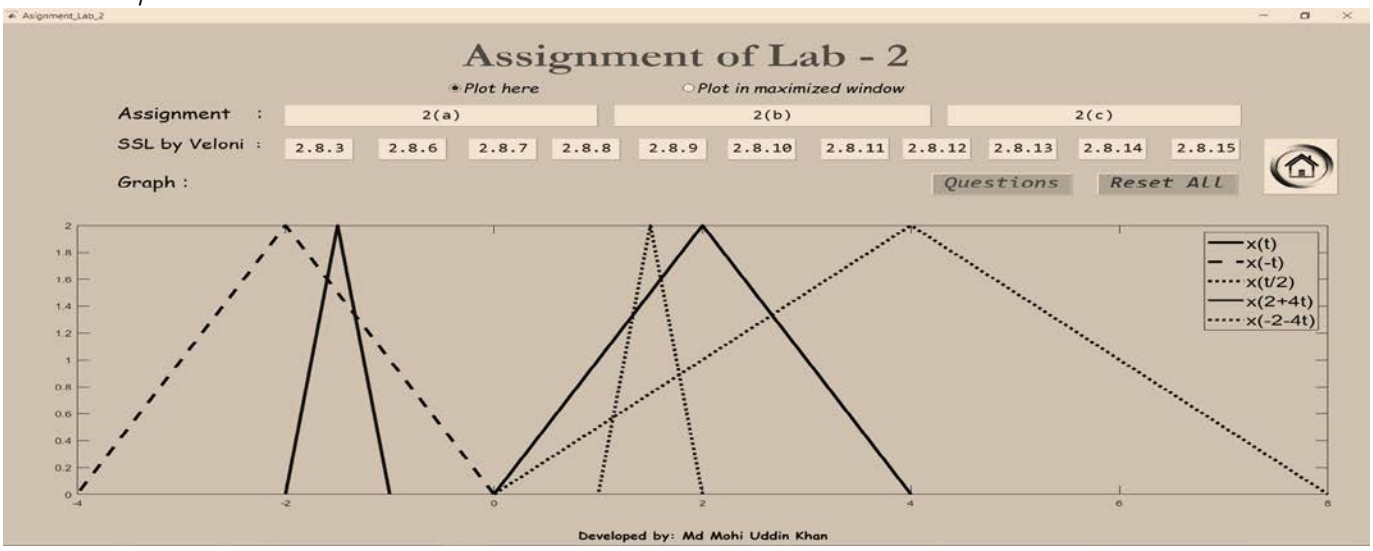
- Assignment 2(c):



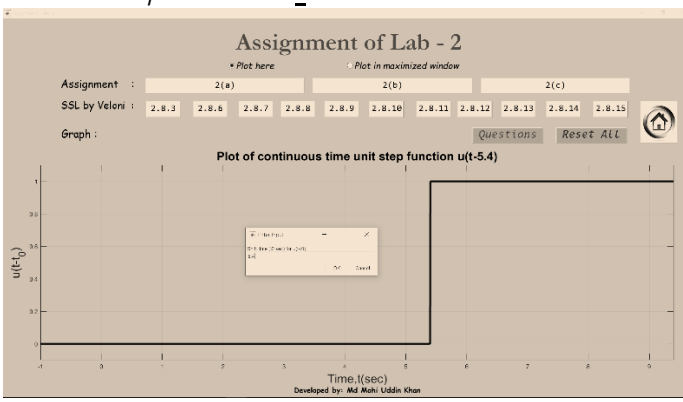
- SSL by Veloni – 2.8.3 :



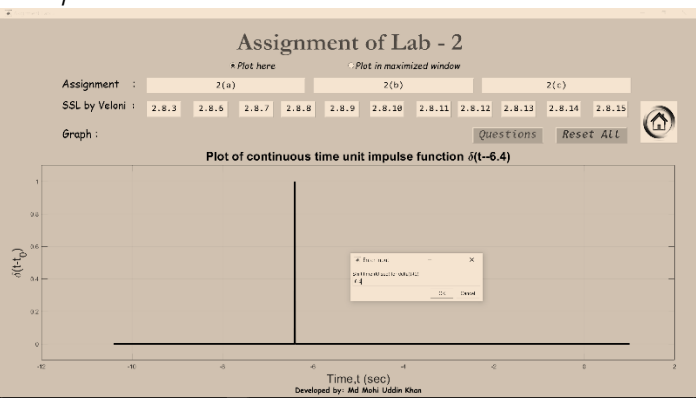
- SSL by Veloni – 2.8.6 :



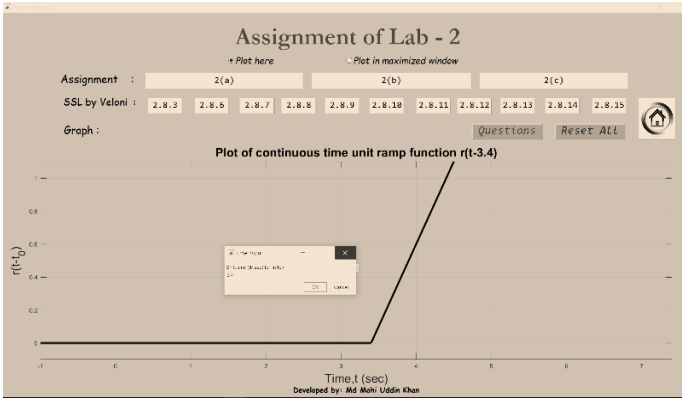
- SSL by Veloni – 2.8.7 :_



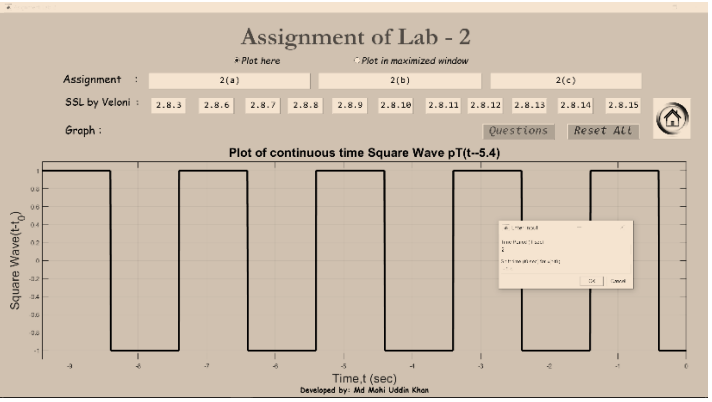
SSL by Veloni – 2.8.8 :



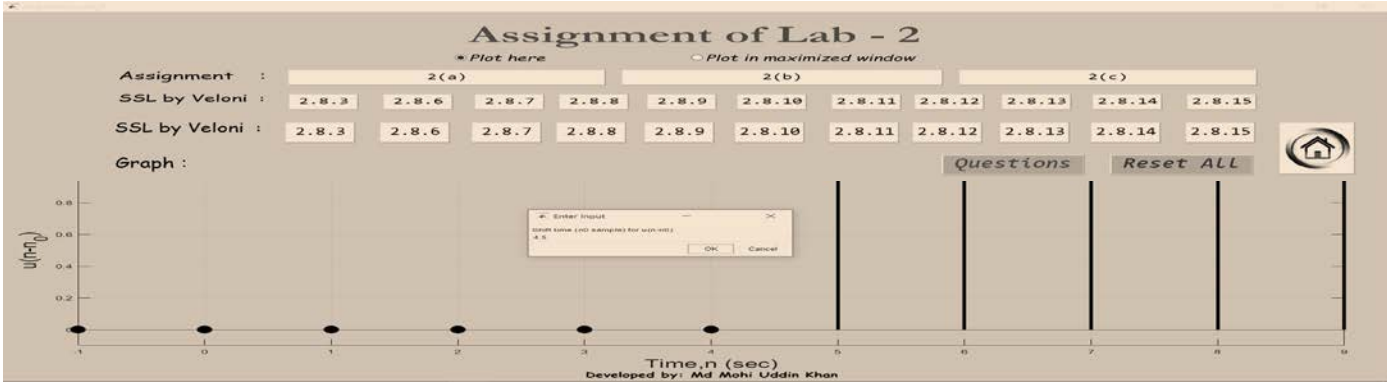
- SSL by Veloni – 2.8.9 :



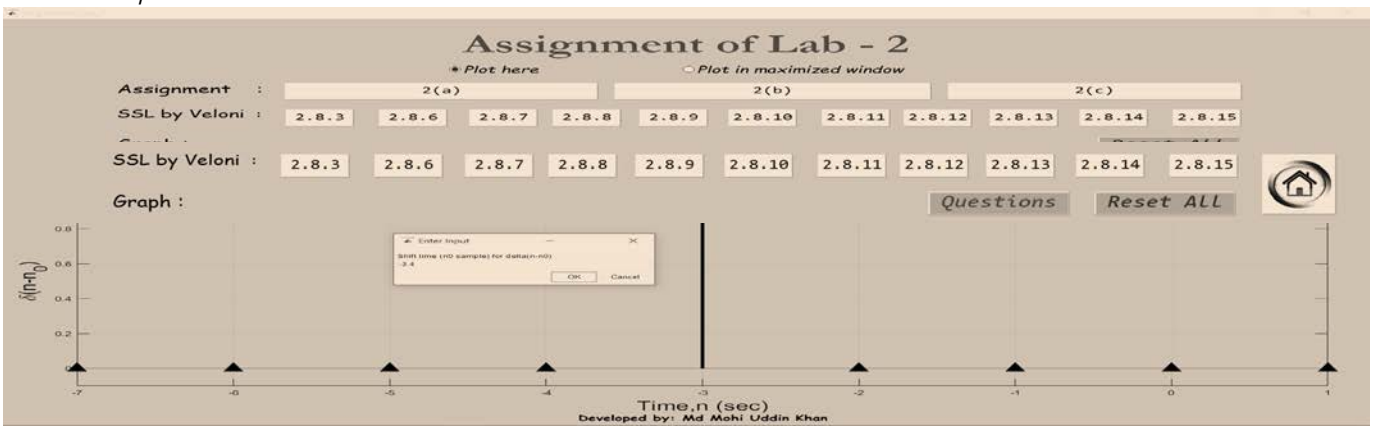
SSL by Veloni – 2.8.10 & 2.8.11:



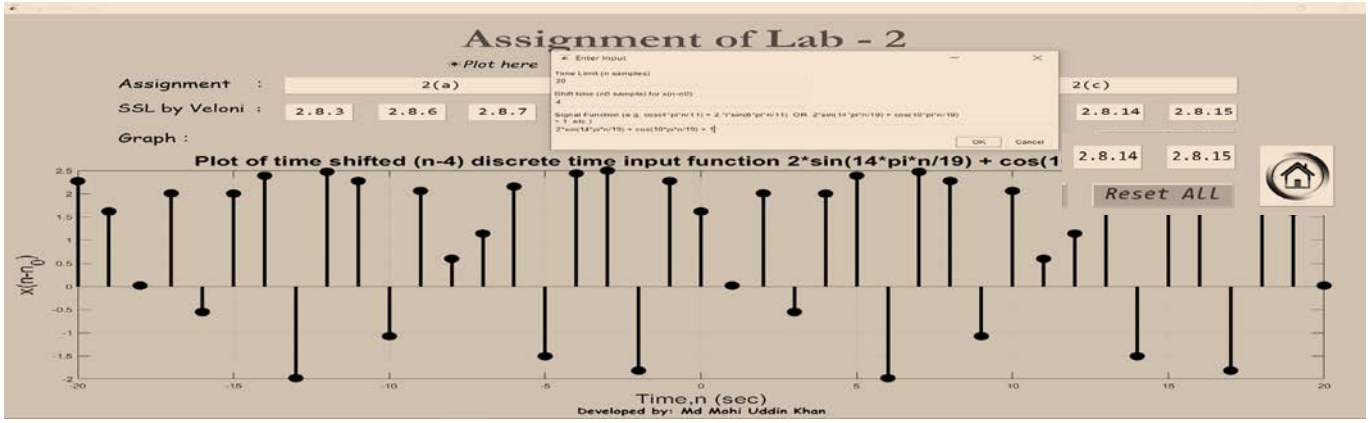
- SSL by Veloni – 2.8.12 :



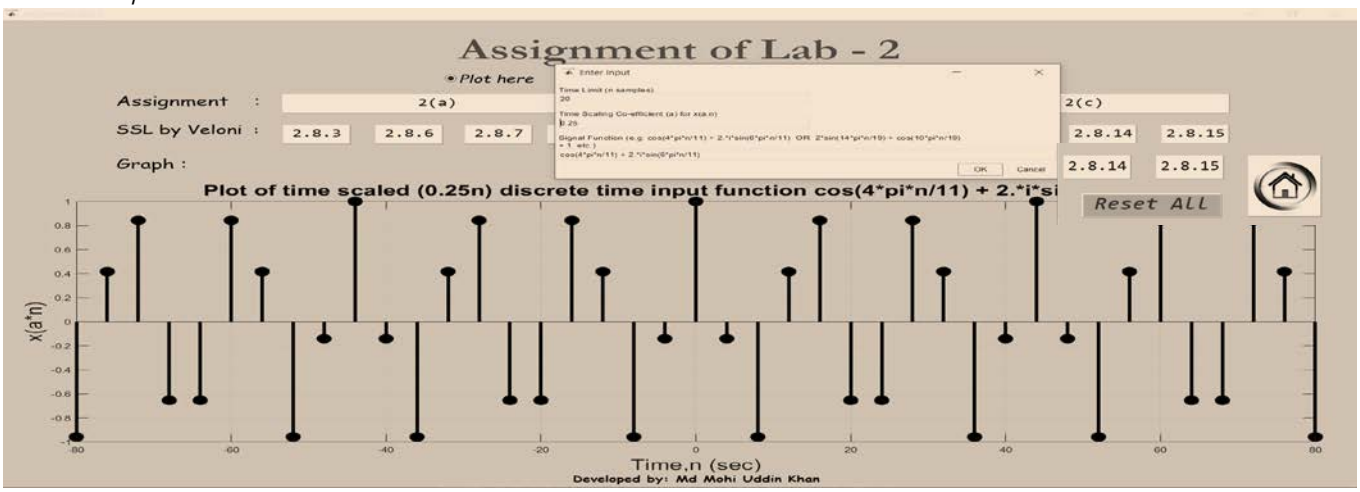
- SSL by Veloni – 2.8.13 :



- SSL by Veloni – 2.8.14 :



- SSL by Veloni – 2.8.15 :



ASSIGNMENT OF LAB – 3 & 4

- GUI Code :

```
function varargout = Assignment_Lab_4(varargin)
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Assignment_Lab_4_OpeningFcn, ...
                   'gui_OutputFcn',  @Assignment_Lab_4_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end

function Assignment_Lab_4_OpeningFcn(hObject, eventdata, handles, varargin)
% Choose default command line output for Assignment_Lab_4
handles.output = hObject;
% Update handles structure
guidata(hObject, handles);

function varargout = Assignment_Lab_4_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;

% Changed here-----------------------------------
% Setting Home button Icon on PushButton
home = imread('Home_icon_(resized 7x7).png');
set(handles.go_home,'CData',home);
%-------------------------------------------------

function Three_Callback(hObject, eventdata, handles)
% to view Axes propeties, type in command window 'uiaxes'
% for more: doc uiaxes
% for more: doc uiaxes properties

% For More: doc inputdlg
input = inputdlg({'Enter pulse width, tau: '},'Input',[1 50]);
% keep showing error if input is empty
flag = 1;
while flag  % means, while flag == 1
    for i=1:size(input,1)  % row size
        if isempty(str2num(input{i}))
            msgbox('Empty input detected. Enter values properly.','Invalid Input','error'); % for more: doc
msgbox
            input = inputdlg({'Enter pulse width, tau: '},'Input',[1 50]);
        else
            if i==size(input,1)
                flag = 0;
            end
        end
    end
end

tau = str2num(input{1});

if get(handles.Maximized_plot,'Value')
    Assignment_3(tau)
else
```

```
    % If Plot_here is selected or no radio button selected, plot will be in GUI Axes
    % Code in 'else' is the same code copied from 'if' section's called function;
    % just coded for plotting in GUI axes
    set(handles.Plot_here,'Value',1);

    % Copied from: Assignment_3(tau)
    % rectangular signal in time domain
    t =  -tau:0.01:tau;
    i = find(abs(t)<tau/2);
    x_t = zeros(size(t));
    x_t(i) = 1;

    % Changed Here
    plot(handles.Graph,t,x_t,'k','LineWidth',3)
    set(handles.Graph,'Color',[0.3 0.75 0.93])
    %
    xlabel('Time (t sec)','FontSize',20);
    ylabel('x(t)','FontSize',20);
    title('Plot of Square Pulse in Time Domain','FontSize',20);
    axis([min(t) max(t) -0.1 1.1]);
    grid on

    % rectangular signal in frequency domain
    x_w = @(tau,omega) tau.*(sin(pi*omega*tau/(2*pi)))./(pi*omega*tau/(2*pi)); % tau*sinc(omega*tau/(2*pi));
    omega = linspace(-4*pi,4*pi,1000);

    pause(8)
    % Changed here
    plot(handles.Graph,omega,x_w(tau,omega),'k','LineWidth',3)
    set(handles.Graph,'Color',[0.3 0.75 0.93])
    %
    xlabel('Frequency (\omega radian)','FontSize',20);
    ylabel('x(\omega)','FontSize',20);
    title('Plot of Square Pulse in Frequency Domain','FontSize',20);
    xlim([min(omega) max(omega)]);
    grid on
end

function Four_b_Callback(hObject, eventdata, handles)
imshow('Openheim_Example_2_3_2_4_Qtn.PNG'); % For more: doc imshow
% to view Axes propeties, type in command window 'uiaxes'
% for more: doc uiaxes
% for more: doc uiaxes properties
if get(handles.Maximized_plot,'Value')
   [nx x nh h msg] = Assignment_4b()
else
    % If Plot_here is selected or no radio button selected, plot will be in GUI Axes
    % Code in 'else' is the same code copied from 'if' section's called function;
    % just coded for plotting in GUI axes
    set(handles.Plot_here,'Value',1);

    % Copied from: [nx x nh h msg] = Assignment_4b()
    % For More: doc inputdlg
input = inputdlg({'alpha for Ex-2.3, where 0 < alpha < 1','alpha for Ex-2.4, where alpha > 1','Time index of
x:   min(n)    max(n)','Time index of h:   min(n)    max(n)'},'Input', [1 50; 1 50; 1 50; 1 50]);

% keep showing error if input is empty
flag = 1;
while flag  % means, while flag == 1
    for i=1:size(input,1)  % row size
        if isempty(str2num(input{i}))
            msgbox('Empty input detected. Enter values properly.','Invalid Input','error'); % for more: doc
msgbox
            input = inputdlg({'alpha for Ex-2.3, where 0 < alpha < 1','alpha for Ex-2.4, where alpha >
```

```
1','Time index of x:   min(n)    max(n)','Time index of h:   min(n)    max(n)'},'Input', [1 50; 1 50; 1 50;
1 50]);
        else
            if i==size(input,1)
                flag = 0;
            end
        end
    end
end

msg = 'Openheim Ex-2.3';
nx = min(str2num(input{3})) : max(str2num(input{3}));
nh = min(str2num(input{4})) : max(str2num(input{4}));
x  = str2num(input{1}).^nx;
h  = nh;
h(:)  = 1;
% Copied from: dconv_160021163(nx,x,nh,h,msg)
grid on;
% changed here
stem(handles.Graph,nx,x,'filled','MarkerSize',7,'color','k','LineWidth', 3)
set(handles.Graph,'Color',[0.3 0.75 0.93])
%
xlabel('Time (n samples)', 'FontSize',15);
ylabel('x[n]', 'FontSize',15);
title([msg,': Plot of x[n]'],'FontSize',20);

pause(5)
grid on;
% changed here
stem(handles.Graph,nh,h,'filled','MarkerSize',7,'color','k','LineWidth', 3)
set(handles.Graph,'Color',[0.3 0.75 0.93])
%
xlabel('Time (n samples)', 'FontSize',15);
ylabel('h[n]', 'FontSize',15);
title([msg,': Plot of h[n]'],'FontSize',20);
% you can directly do convolution using built in function
% y = conv(h,x);  % For more: doc conv    & doc deconv
ny = min(nx) + min(nh) : max(nx) + max(nh);
y = zeros(1,length(ny));
z = zeros(1,length(ny));

for i = 1:length(nx)
    y(i) = y(i) + x(i);
end
for i = 1:length(nh)
    z = z + h(i)*y;
    y = circshift(y,1);  % doc circshift
end
y = z;

pause(5)
% Changed here
stem(handles.Graph,ny,y,'filled','MarkerSize',7,'color','k','LineWidth', 3)
set(handles.Graph,'Color',[0.3 0.75 0.93])
%
xlabel('Time (n samples)', 'FontSize',15);
ylabel('y[n]', 'FontSize',15);
title([msg,': Plot of convoluted signal y[n] = x[n]*h[n]'],'FontSize',20);
grid on;
pause(5)
msg = 'Openheim Ex-2.4';
nx = min(str2num(input{3})) : max(str2num(input{3}));
nh = min(str2num(input{4})) : max(str2num(input{4}));
h  = str2num(input{2}).^nh;
```

```
x  = nx;
x(:)  = 1;
% Copied from: dconv_160021163(nx,x,nh,h,msg)
grid on;
% changed here
stem(handles.Graph,nx,x,'filled','MarkerSize',7,'color','k','LineWidth', 3)
set(handles.Graph,'Color',[0.3 0.75 0.93])
%
xlabel('Time (n samples)', 'FontSize',15);
ylabel('x[n]', 'FontSize',15);
title([msg,': Plot of x[n]'],'FontSize',20);

pause(5)
grid on;
% changed here
stem(handles.Graph,nh,h,'filled','MarkerSize',7,'color','k','LineWidth', 3)
set(handles.Graph,'Color',[0.3 0.75 0.93])
%
xlabel('Time (n samples)', 'FontSize',15);
ylabel('h[n]', 'FontSize',15);
title([msg,': Plot of h[n]'],'FontSize',20);

% you can directly do convolution using built in function
% y = conv(h,x);  % For more: doc conv    & doc deconv
ny = min(nx) + min(nh) : max(nx) + max(nh);
y = zeros(1,length(ny));
z = zeros(1,length(ny));

for i = 1:length(nx)
    y(i) = y(i) + x(i);
end
for i = 1:length(nh)
    z = z + h(i)*y;
    y = circshift(y,1);  % doc circshift
end
y = z;

pause(5)
% Changed here
stem(handles.Graph,ny,y,'filled','MarkerSize',7,'color','k','LineWidth', 3)
set(handles.Graph,'Color',[0.3 0.75 0.93])
%
xlabel('Time (n samples)', 'FontSize',15);
ylabel('y[n]', 'FontSize',15);
title([msg,': Plot of convoluted signal y[n] = x[n]*h[n]'],'FontSize',20);
grid on;
end

function Four_c_Callback(hObject, eventdata, handles)
% For More: doc inputdlg
input = inputdlg({'Enter Function: sin(x) or cos(x) or tan(x) etc','Enter some integers: m1 m2 m3
...'},'Input', [1 50;1 50]);

% keep showing error if input is empty
flag = 1;
while flag  % means, while flag == 1
    for i=1:size(input,1)  % row size
        if isempty(str2num(input{2}))
            msgbox('Empty input detected. Enter values properly.','Invalid Input','error'); % for more: doc
msgbox
            input = inputdlg({'Enter Function: sin(x) or cos(x) or tan(x) etc','Enter some integers: m1 m2
m3 ...'},'Input', [1 50;1 50]);
        else
            if i==size(input,1)
```

```
                flag = 0;
            end
        end
    end
end

fn = inline(input{1},'x');  % trigonometric function
m = str2num(input{2});        % Integers

if get(handles.Maximized_plot,'Value')
    Assignment_4c(input,fn,m)
else
    % If Plot_here is selected or no radio button selected, plot will be in GUI Axes
    % Code in 'else' is the same code copied from 'if' section's called function;
    % just coded for plotting in GUI axes
    set(handles.Plot_here,'Value',1);

    % Copied from: Assignment_4c(input,fn,m)
    n = -30:30;  % Time Index
    N = 5;       % Period
    F = 1/N;     % Fundamental frequency
    omega = 2*pi*F;

for k = 1:numel(m)
    y_time_domain = fn((omega + 2*pi*m(k))*n);
    y_freq_domain = abs(fft(y_time_domain));
    % Changed here
    stem(handles.Graph,n,y_time_domain,'filled','MarkerSize',7,'LineWidth', 3,'color','k')
    set(handles.Graph,'Color',[0.3 0.75 0.93])
    %
    xlabel('Time (n samples)', 'FontSize',15);
    ylabel(input{1}, 'FontSize',15);
    title(['Plot of ',input{1},' in time domain, where x = (2\piF + 2\pi',
num2str(m(k)),')n'],'FontSize',20);
    grid on
    hold on
    pause(3)
end

hold off

for k = 1:numel(m)
    y_time_domain = fn((omega + 2*pi*m(k))*n);
    y_freq_domain = abs(fft(y_time_domain));
    % Changed here
    stem(handles.Graph,y_freq_domain,'filled','MarkerSize',7,'LineWidth', 3,'color','k')
    set(handles.Graph,'Color',[0.3 0.75 0.93])
    %
    xlabel('Frequency (Hz)', 'FontSize',15);
    ylabel(input{1}, 'FontSize',15);
    title(['Plot of ',input{1},' in frequency domain, where x = (2\piF + 2\pi',
num2str(m(k)),')n'],'FontSize',20);
    grid on
    hold on
    pause(3)
end
xlabel('Therefore, Discrete time sinusoids, whose frequencies are separated by an integer multiple of 2\pi
are identical.','FontSize',20);

hold off

end
```

```
function Four_d_Callback(hObject, eventdata, handles)
% For More: doc inputdlg
input = inputdlg({'Enter Function: sin(x) or cos(x) or tan(x) etc','Enter a rational number: p/q
format'},'Input', [1 50;1 50]);

% keep showing error if input is empty
flag = 1;
while flag  % means, while flag == 1
    for i=1:size(input,1)  % row size
        if isempty(str2num(input{2}))
            msgbox('Empty input detected. Enter values properly.','Invalid Input','error'); % for more: doc
msgbox
            input = inputdlg({'Enter Function: sin(x) or cos(x) or tan(x) etc','Enter a rational number: p/q
format'},'Input', [1 50;1 50]);
        else
            if i==size(input,1)
                flag = 0;
            end
        end
    end
end

fn = inline(input{1},'x');  % trigonometric function
m = str2num(input{2});        % rational number of p/q format


if get(handles.Maximized_plot,'Value')
    Assignment_4d(input,fn,m)
else
    % If Plot_here is selected or no radio button selected, plot will be in GUI Axes
    % Code in 'else' is the same code copied from 'if' section's called function;
    % just coded for plotting in GUI axes
    set(handles.Plot_here,'Value',1);

    % Copied from: Assignment_4d(input,fn,m)
    n = -60:60;  % Time Index

    F = m;
    y_time_domain = fn(2*pi*F*n);
    % Changed here
    stem(handles.Graph,n,y_time_domain,'filled','MarkerSize',7,'LineWidth', 3,'color','k')
    set(handles.Graph,'Color',[0.3 0.75 0.93])
    %
    xlabel('Time (n samples)', 'FontSize',15);
    ylabel(input{1}, 'FontSize',15);
    title(['Plot of ',input{1},' in Time domain, where x = (2\piF)n  &  F = ', num2str(m),' is rational
number'],'FontSize',20);
    legend('Periodic Signal');
    grid on

    pause(7)
    y_time_domain1 = fn(m.*n);
    % Changed here
    stem(handles.Graph,n,y_time_domain1,'filled','MarkerSize',7,'LineWidth', 3,'color','k')
    set(handles.Graph,'Color',[0.3 0.75 0.93])
    %
    xlabel('Time (n samples)', 'FontSize',15);
    ylabel(input{1}, 'FontSize',15);
    title(['Plot of ',input{1},' in Time domain, where x = mn  &  F = ', num2str(m),'/2\pi is irrational
number'],'FontSize',20);
    legend('Aperiodic Signal');
    grid on

pause(7)
```

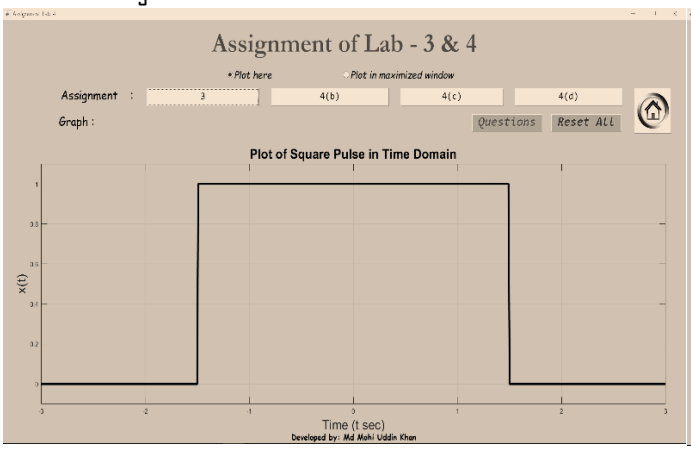
```
xlabel('Carefully notice in 2nd plot using figure "data cursor" tool in Maximized window plot: y-axis values
are not equal after certain interval.','FontSize',17);
pause(7)
xlabel('Therefore, a Discrete time sinusoid is periodic if its frequency is rational
number.','FontSize',20);

end

function Plot_here_Callback(hObject, eventdata, handles)
set(handles.Maximized_plot,'Value',0);

function Maximized_plot_Callback(hObject, eventdata, handles)
set(handles.Plot_here,'Value',0);

function View_Questions_Callback(hObject, eventdata, handles)
% for more type in command window 'doc figure' , 'doc Figure Properties'
f = figure('Name','Assignment-3 & 4 Questions','NumberTitle','off');
f.WindowState = 'maximized';
f.WindowStyle = 'docked';
% Docked window can't be maximized. In normal MATLAB, docked window is
% nice to look but in executable app there is no docked window mode. So I
% also need maximized command for .exe app. So, both commands are used &
% maximized window declared first since docked window can't be maximized.
imshow('Qtn_Assignment_3_4.png');   % for more: doc imshow

function Reset_All_Callback(hObject, eventdata, handles)
clc
evalin('base','clear all');  % clears the variables under current GUI. i.e: base gui
Close_figures_except_GUI()

% Clearing UIAXES
plot(handles.Graph,0,0)
set(handles.Graph,'Color',[0.3 0.75 0.93])
grid on
%
set(handles.Plot_here,'Value',0);
set(handles.Maximized_plot,'Value',0);

function go_home_Callback(hObject, eventdata, handles)
clc
evalin('base','clear all');  % clears the variables under current GUI. i.e: base gui
Close_figures_except_GUI()

delete(handles.Lab_4_GUI) % closes current GUI
cd 'E:\IUT Books\6th Semester EEE\LAB\EEE 4602_Signals & Systems\Lab 5_Home_page_for_GUI'
Homepage();
```

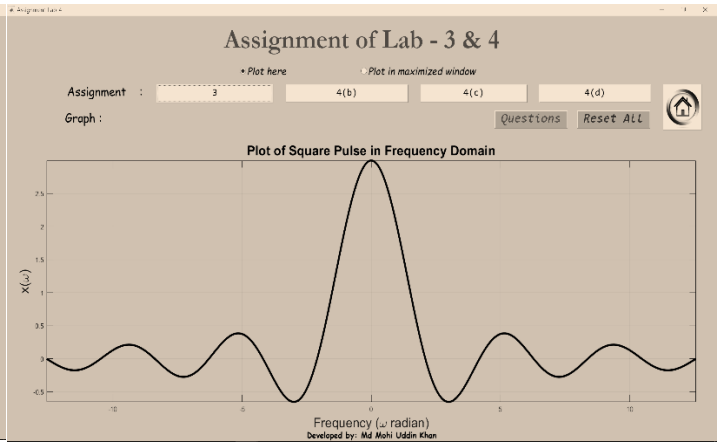
- Code of external function called into GUI code:

```
function Close_figures_except_GUI()
% Detect all figures - close the figures that are not the GUI
fh=findall(0,'type','figure');
% OR [you can use either of fh]
fh=findobj(0,'type','figure');
nfh=length(fh); % Total number of open figures, including GUI and figures with visibility 'off'
% Scan through open figures - GUI figure number is [] (i.e. size is zero)
for i = 1:nfh
    % Close all figures with a Number size is greater than zero
    if sum(size(fh(i).Number)) > 0
        figure(fh(i).Number);
        close(gcf);
    end
end
end
```

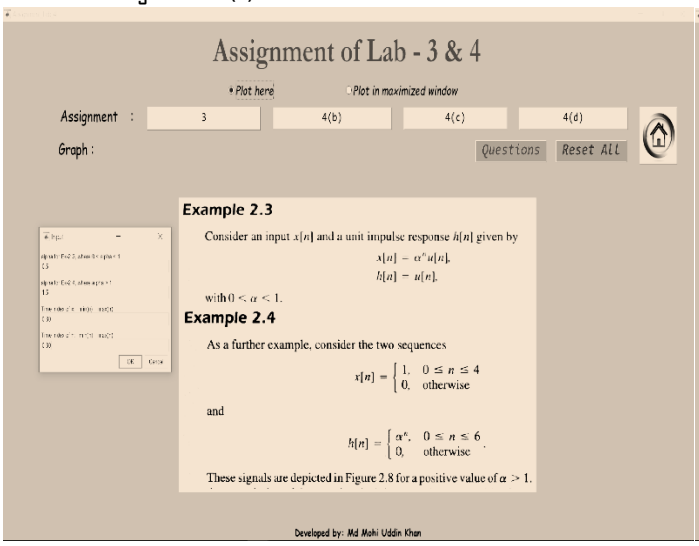
Output Graphs

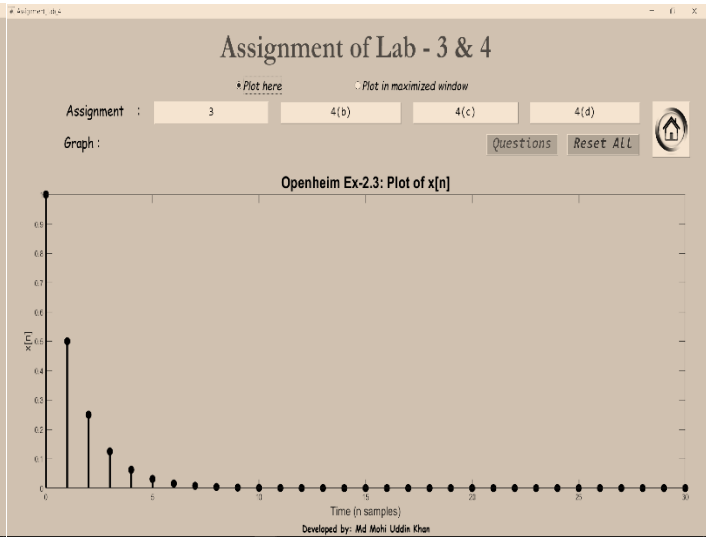
- Assignment 3:

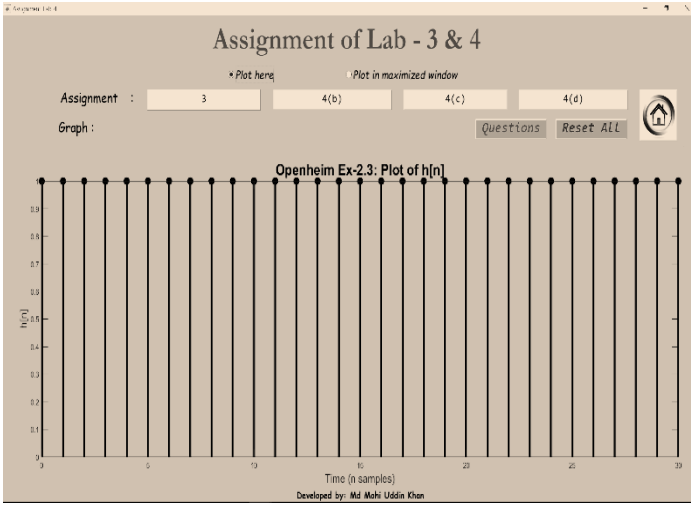


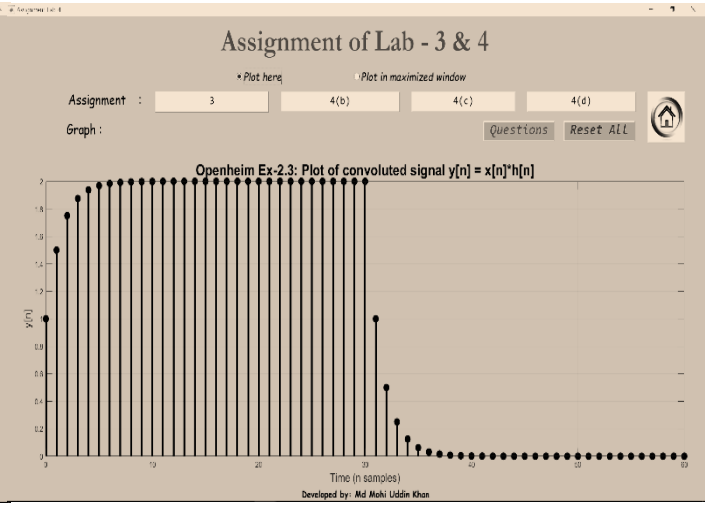


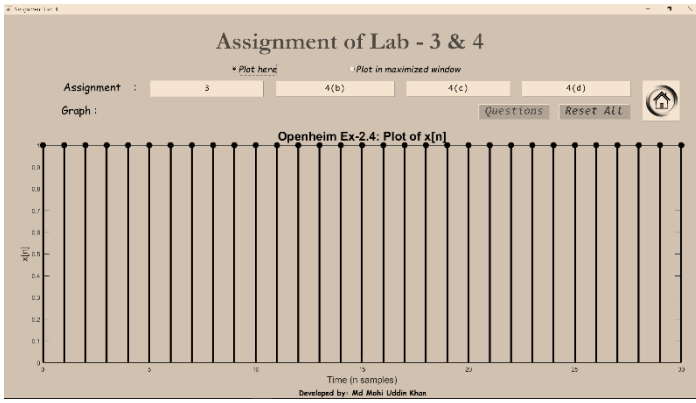
- Assignment 4 (b):

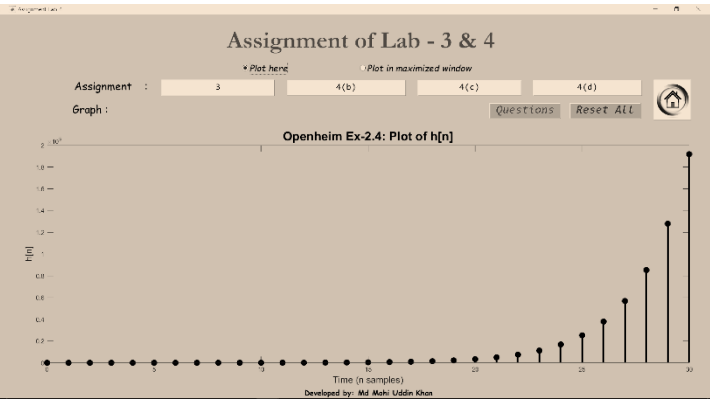


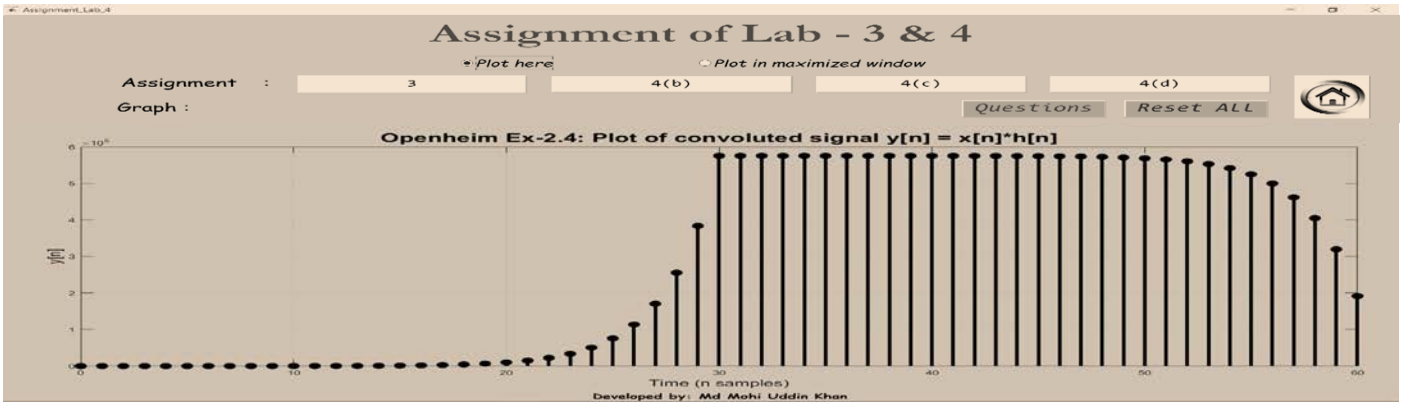




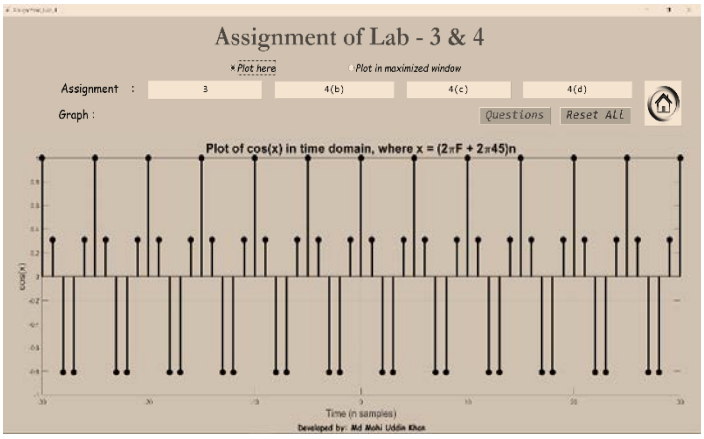


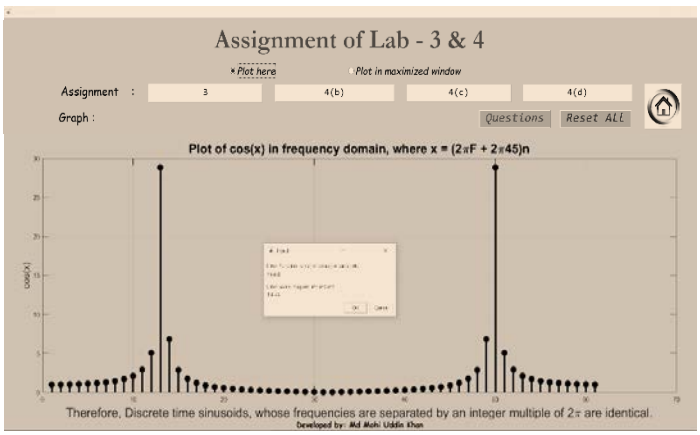






- Assignment 4(c):





- Assignment 4(d):