```
diary on
format compact
%Johnny Li
%EEL3135    Fall 2018
%Lab 0 Part 1

%1.1.1
6^2
ans =
    36
%The value of the function 6^2 was calculated and stored in the default
%variable, ans, with the result value of 36.
ans/6
ans =
     6
%Next the variable ans, with the value of 36, was divided by 6. The variable
%ans was then updated with the new result of 6.
ans
ans =
     6
%The variable ans was called and displayed the value of 6.

%1.1.2
pi*pi-10
ans =
   -0.1304
%The function pi*pi-10 was calculated, with pi equaling the stored value of
%pi in MATLAB library, and the result- approximating to -0.1304- was stored
%in the default variable ans.
sin(pi/4)
ans =
    0.7071
%The function of sin(pi/4) was calculated, with the trigonometry function of
%sin(x), and the result approximating to 0.7071 which replaced the stored
%value of ans.

ans^2
ans =
    0.5000
%The stored value, equaling to 0.7071, of the variable ans was squared which
%resulted in the approximation of 0.5. The value of ans was updated to the
%new result, 0.5000.

%1.1.3
x=sin(pi/5)
x =
    0.5878
%Create the variable x and store the result of the function sin(pi/5) in the
%variable, being 0.5878.
cos(pi/5)
ans =
    0.8090
%The function cos(pi/5) was calculated to 0.8090 and was stored in the
%default variable - ans.
y=sqrt(1-x*x)
y =
```

```
    0.8090
%Create the variable y and store the result of the function. The function
%called for the variable x where the stored value was retrieved and used in
%the calculation. The result came to be 0.8090.
ans
ans =
    0.8090
%The variable ans was called and displayed the current stored value of
%0.8090.

%Question: What is the numerical value of x and y?
%x=0.5878 and y=0.8090

%1.2.1
x=0:6
x =
     0     1     2     3     4     5     6
%Create the variable x and stored a 1x7 matrix with the value of 0 to 6
%with an incrementation of 1. Each value is stored in their own respective
%index in the matrix.
y=2:4:17
y =
     2     6    10    14
%Create the variable y and stored a 1x4 matrix with the starting value of 2,
%stated in the first number, to the ending value of 17, stated in the last
%value.
%Because the incrementation was by 4, stated by the middle value, the final
%matrix value closest to 17 was 14 since 17 cannot be reached by incrementing
%4 from the initial value of 2.
z=2:(1/9):4
z =
  Columns 1 through 10
    2.0000    2.1111    2.2222    2.3333    2.4444    2.5556    2.6667
2.7778    2.8889    3.0000
  Columns 11 through 19
    3.1111    3.2222    3.3333    3.4444    3.5556    3.6667    3.7778
3.8889    4.0000
%Create the variable z and stored a 1x19 matrix with the starting value of 2,
%stated in the first number (2), to the ending value of 4, stated in the last
%value (4).
%The incrementation was by 1/9 as stated by the middle value. The resulting
%value was estimated due to rounding therefore this created some variation in
%the ending decimal.

t=pi*[0:0.1:2]
t =
  Columns 1 through 10
         0    0.3142    0.6283    0.9425    1.2566    1.5708    1.8850
2.1991    2.5133    2.8274
  Columns 11 through 20
    3.1416    3.4558    3.7699    4.0841    4.3982    4.7124    5.0265
5.3407    5.6549    5.9690
  Column 21
    6.2832
%Create the variable t and stores the result of the function, pi multiplied
%with a 1*21 matrix with the starting value of 0, stated in the first number
```

```
%(0), to the ending value of 2, stated in the last value (2). The
%incrementation was by 0.1 as stated by the middle value.
%Therefore, the stored value in the variable t was the pi multiplied the
%constructed matrix, resulting in t storing a 1x21 matrix.

%1.2.2
xx = [zeros(1,3), linspace(0,1,5), ones(1,4)]
xx =
  Columns 1 through 10
         0         0         0         0    0.2500    0.5000    0.7500
1.0000    1.0000    1.0000
  Columns 11 through 12
    1.0000    1.0000
%Create the variable xx and stores a 1x12 vector. The zeros(1,3) creates a
%1x3 vector with all zeros. These values are stored in the first three
%indexes of the xx variable vector, as stated in (1,3).
%The linspace(0,1,5) generate a linearly spaced vector starting at the inital
%value of 0 and ending at the value of 5.
%The vector spacing between the points is based on the function (x2-x1)/(n-
%1). Where x1=0, x2=1, and n=5. This is similar to the : command but with
%more control.
%This vector is stored in the next set of indexes of the xx-variable vector.
%The ones(1,4) creates 1x4 vector with all ones. This vector encompass the
%the last indexes of the xx-variable vector.
xx(4:6)
ans =
         0    0.2500    0.5000
%The value of index 4 to 6 in the variable xx vector was called and
%displayed.
size(xx)
ans =
     1    12
%Display the number of rows, initial value (1), and the number of columns,
%final value (12), of the xx-variable vector.
length(xx)
ans =
    12
%Display the max length value (12), number of indexes, of the xx variable
%vector.
xx(2:2:length(xx))
ans =
         0         0    0.5000    1.0000    1.0000    1.0000
%Call and display the value of the index starting at the initial value of 2
%to the length(xx), max length of xx variable vector (12), in the
%incrementation of 2.
yy = xx
yy =
  Columns 1 through 9
         0         0         0         0    0.2500    0.5000    0.7500
1.0000    1.0000
  Columns 10 through 12
    1.0000    1.0000    1.0000
%Create the variable yy equal to the variable xx.
yy(4:6) = pi*(1:3)
yy =
  Columns 1 through 9
```

```
         0         0         0    3.1416    6.2832    9.4248    0.7500
1.0000    1.0000
  Columns 10 through 12
    1.0000    1.0000    1.0000
%Retrieved the values in the index of 1 to 3 of the yy variable vector
%and multiple by the value of pi and store the results back into the
%index of 4 to 6 of the yy variable vector. The values in the index
%of 1 to 3 of the yy variable vector remain unchanged.

%1.2.3
xx(2:2:length(xx))=pi^pi
xx =
  Columns 1 through 9
         0   36.4622         0   36.4622    0.2500   36.4622    0.7500
36.4622    1.0000
  Columns 10 through 12
   36.4622    1.0000   36.4622
%Take the vector xx and take the elements with an even index {xx(2), xx(4),
%...} and replace them with pi^pi, which is 36.4622.

%1.3.1
x=0;
for k = 1:4
x = x+1;
end
%This was a loop command where a variable x was create and
%stored the initial value of zero and incremented it by 1,
%having the value of x add by 1 and store it back in the
%variable x, for each run of k, start at one and end at
%zero (4 times).

%Question: What's the value of x after the code executes?
%x=4

%1.3.2
t=zeros(1,9);
for n=1:9
t(n)=n^n;
end
%Created t variable vector for storage. Use a for loop from 1 to 9,
%to generate vector of 1^1 to 9^9 based on n (the number of run).


%1.3.3

%Question: What is the purpose of the dot before the asterisk in line two of
the above?
%The dot symbolizes that the multiplication is an element-wise, or pointwise,
%operation which means multiplying the image of the two functions, x and y,
%at each value in the domain.


xk=(1:200)/50)
rk = sqrt(xk.*xk + 2.25);
sig = exp(j*2*pi*rk);
plot(xk,sig,'mo-');
```

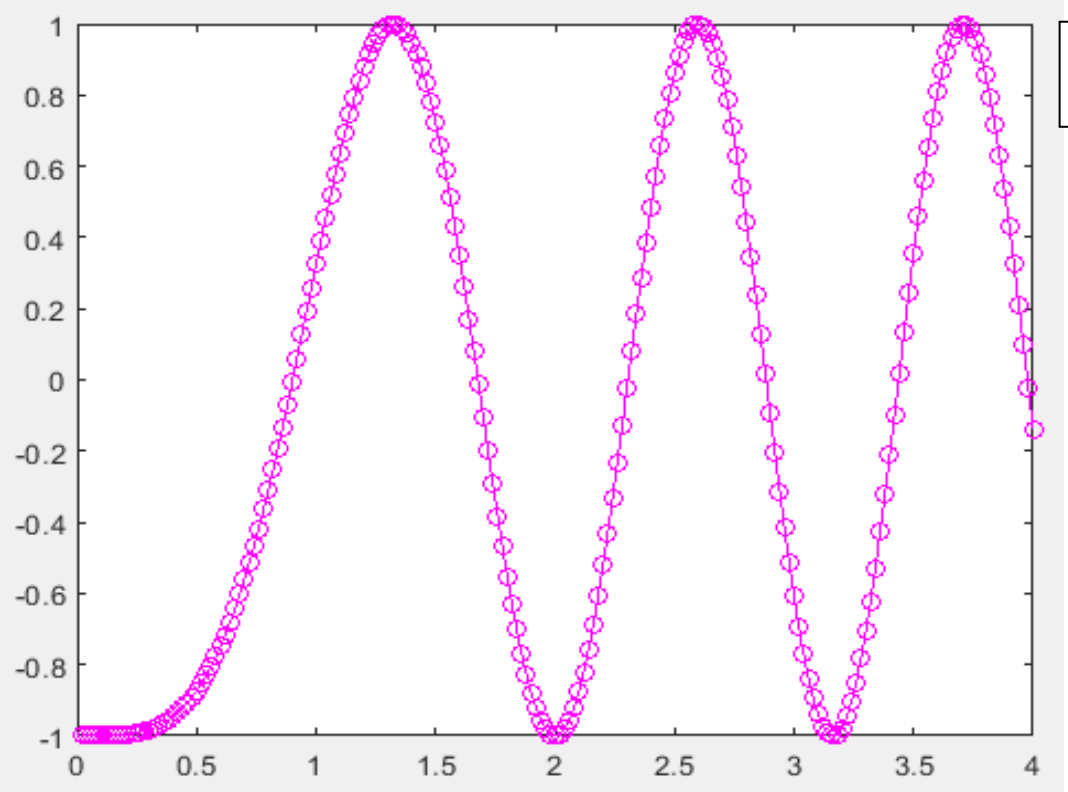

Figure 1: 1.3.3 plot
`plot(xk,sig,'mo-')`

```
%The code given was optimized by removing the for loop and using the element-
%wise multiplication (.*) to have the function multiple each element.
%The function was then plotted as shown above.

%1.4
x = [-3 -1 0 1 3];
y = x.*x - 3*x;
%A variable vector x was created and stored the values -3 to 3. A variable
%vector y was created and stored the multiplication of vector x by itself
%subtracted by 3 times the x vector.
plot(x, y)
%The values of x and y was plotted respectively, as shown below.
```

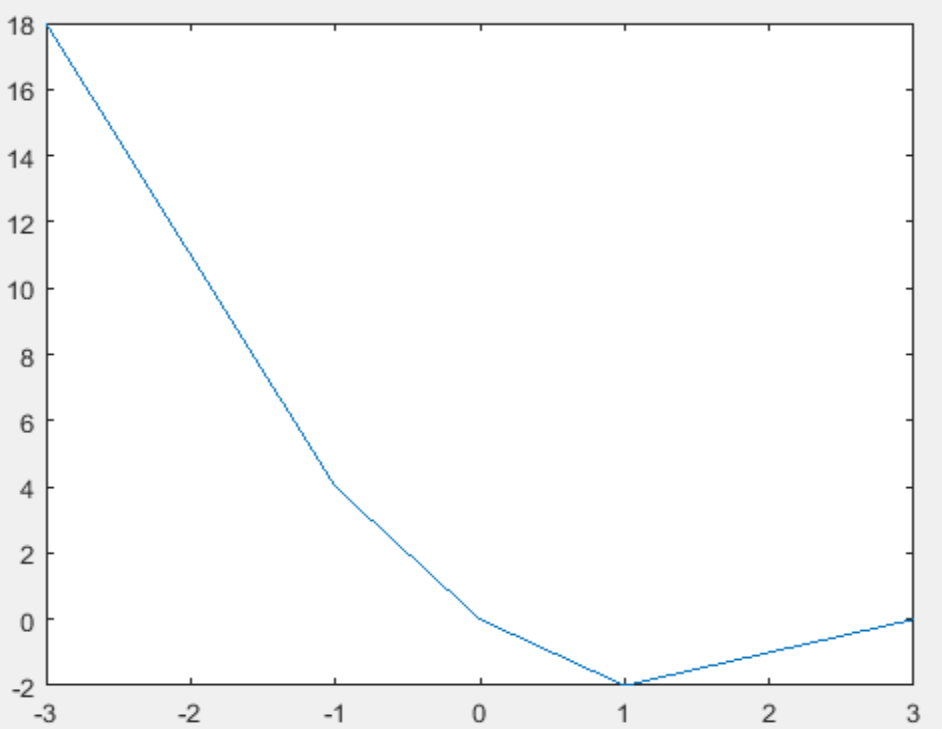

Figure 2: 1.4
plot(x,y)

```
z = x + y * sqrt(-1)

z =

  -3.0000 +18.0000i  -1.0000 + 4.0000i   0.0000 + 0.0000i   1.0000 - 2.0000i
3.0000 + 0.0000i

%The values of vector x are added with the product of the values of vector y
%times sqrt(-1), which is an imaginary number hence the i and stored in the
%variable z.
plot(z)
%The plot of z is the same as the plot(x,y) however the y-axis is the
%imaginary plane since they have the same positions when plotted.

%Question: What is the difference between a*b and a.*b, where a and b are
matrices?
%The function a*b would multiple the matrices by their dimensions in a dot
%product fashion. While a.*b would multiple each element in matrix a by their
%respective counterpart element in matrix b and then form an new matrix with
%the results.

%1.5.1
type oddsummer

function s = oddsummer(even)
%ODDSUMMER: Takes a positive integer n as its argument, and returns the
%sum of all odd numbers between 1 and n.

%Error Messages
%msg1('The input was not positive.');
```

```
%msg2('The input was not a integer.');
%msg3('The input was not even.');

if(even<0) || (even==0)
    message('The input was not positive.');
%Check if the input is positive, if not return error.

elseif abs(even-floor(even))>0
    message('The input was not a integer.');
%Check if the input is a integer, if not return error.

elseif mod(even,2)==1
    message('The input was not even.');
%Check if the input is even, if not return error.

else
odds=1:2:even;
%Store all odd numbers in a vector.
s=sum(odds);
%Sum all odd numbers.
end

end

oddsummer(5)
Invalid Message ID format: 'The input was not even.'

    message('The input was not even.');
}_
oddsummer(4)

ans =

     4

oddsummer(2.5)
Invalid Message ID format: 'The input was not a integer.'

    message('The input was not a integer.');
}_
oddsummer(10)

ans =

    25

oddsummer(-7)
Invalid Message ID format: 'The input was not positive.'

    message('The input was not positive.');
}
```

```
%1.5.2
type hellos

function h= hellos(n)
%HELLOS: Taking a positive integer n as its argument and displays the
%word hello n times.

if(n<0) || (n==0)
    message('The input was not positive.');
%Check if the input is positive, if not return error.

elseif abs(n-floor(n))>0
    message('The input was not a integer.');
%Check if the input is a integer, if not return error.

else
    for h=1:n
        fprintf('%d. hello\n', h);
%Output hello n times.
    end
end
end

hellos(1)
1. hello

hellos(2.5)
Invalid Message ID format: 'The input was not a integer.'.

    message('The input was not a integer.');
}_
hellos(-5)
Invalid Message ID format: 'The input was not positive.'

    message('The input was not positive.');
}_
hellos(3)
1. hello
2. hello
3. hello

hellos(6)
1. hello
2. hello
3. hello
4. hello
5. hello
6. hello

diary off
```