1.

a) x=[3 7 -1 14] is setting the elements in the vector equal to x. The output is confirming that x is now equal to the vector that was set.

b) size(x) is to find the dimensions of a vector. In this case, since ‘x’ was set to be equal to [3 7 -1 14] in the previous question, the size is 1x4 meaning the dimensions of the vector are 1 row by 4 columns.

c) x(2) is to find the second entry in the vector. In this case, the second number in the vector ‘x’ is 7.

d) y=1:0.2:2 means to set ‘y’ equal to numbers 1 through 2 with a step of 0.2. So, the output shows as 1, 1.2, 1.4 and so on until the number 2.

2.

```
A =

     1     2
     3     4


B =

     1     0     2
     0     3     1
```

a) (i) A*B multiplies the two matrices, A and B.

```
multiply =

     1     6     4
     3    12    10
```

(ii) B’ creates a transpose of the matrix B. The dimensions of the original matrix B is flipped. Instead of 2x3, it is now 3x2.

```
transpose =

     1     0
     0     3
     2     1
```

(iii) A(2,:) finds the second row of the matrix A. Hence, the output below since it is the second row of matrix A.

row =

    3   4

(iv) inv(A) finds the inverse of matrix A. It works because it is a square matrix hence the output below.

invA =

```
  -2.0000    1.0000
   1.5000   -0.5000
```

(v) Since matrix B is not a square matrix, an error occurs when trying to find the inv(B) or inverse of matrix B.

Error using inv
Matrix must be square.

Error in HW1_Problem2and3 (line 22)
invB=inv(B)

b) y = [4/3 1.2345e-6]

format short: produces values rounded to the 4th decimal place
y =

   1.3333   0.0000

format long: produces values to the 15th decimal place
y =
  1.333333333333333   0.000001234500000

format short e: produces values in scientific notation
y =
  1.3333e+00   1.2345e-06

format rat: produces values in fraction form
y =

4/3   1/810045

c)

linspace(1,6,10) displays 10 evenly spaced points between numbers 1 and 6.

```
x =

  Columns 1 through 9

    1.0000    1.5556    2.1111    2.6667    3.2222    3.7778    4.3333    4.8889    5.4444

  Column 10

    6.0000
```

length(x) displays the number of points in x, which is 10.

```
xlength =

    10
```

The for loop computes the log of each of the points contained in x, which is stored int the variable ‘z’ .

```
z=
  Columns 1 through 9

         0    0.4418    0.7472    0.9808    1.1701    1.3291    1.4663    1.5870    1.6946

  Column 10

    1.7918
```

d)
```
if(a>0)
    log(a)
elseif(a<=0)
    disp("log(a) is undefined")
end
```

```
a =

     2
```

ans =

0.6931

a =

-1

log(a) is undefined

3.

a) f =

function_handle with value:

@(x)x.^2-3*x+5

>> f(3)

ans =

5

func.m file contains:

```
function f = func(x)
    f = x.^2 - 3*x +5;
end
```

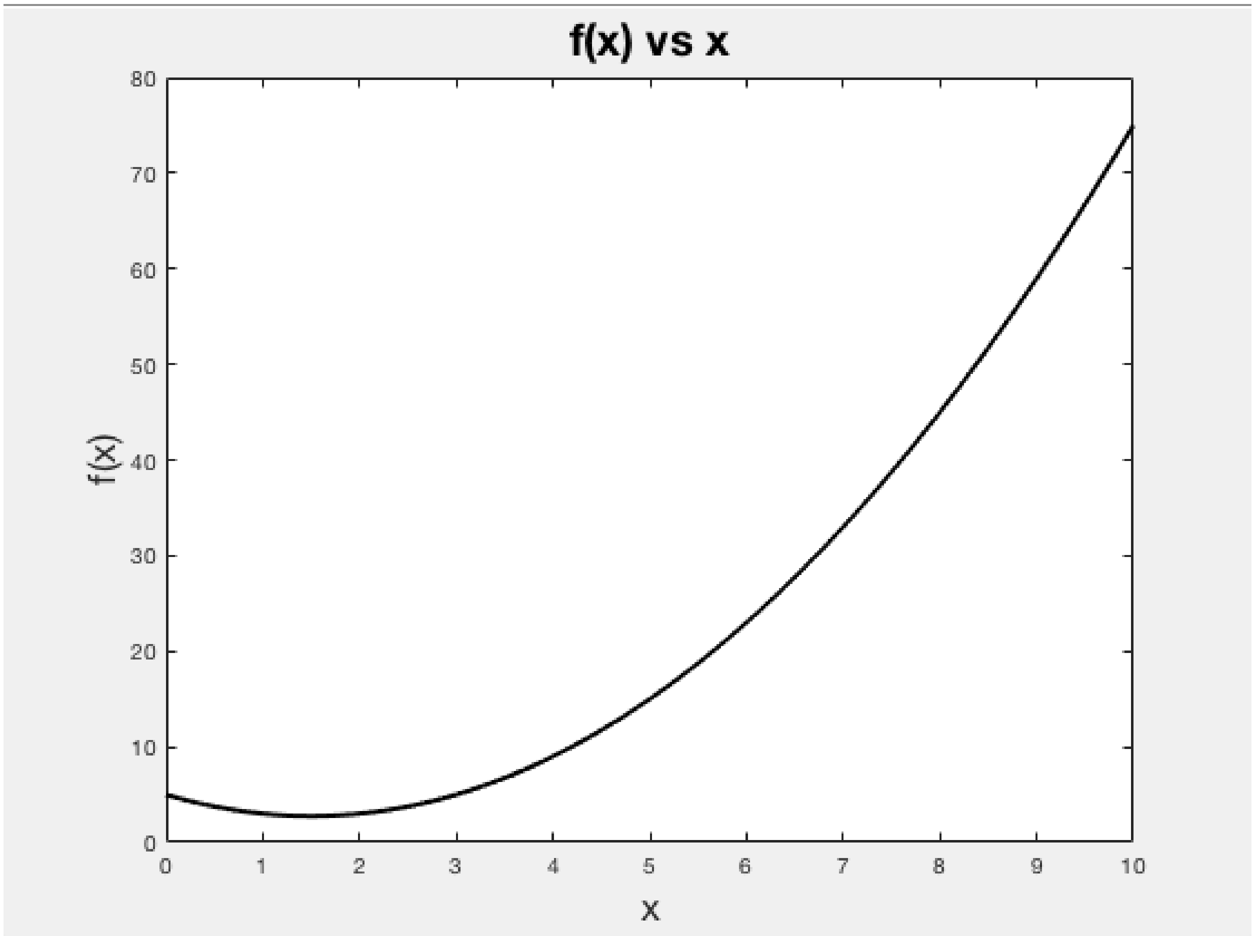
>> func(3)

ans =

5

The two implementations are different in that the first one is directly inputted into the script file while the other is put into a separate file and the function name needs to be called. Both methods give the same answer as 5 when x = 3. The purpose of using .^ command instead of ^ when implementing x^2 is because .^ is for element-wise power in which each element in a vector or matrix is squared while ^ is for the element as a whole.

b) Please see the plot on the next page.



4. 3.1415926
   normalized floating point decimal form: 0.31415926

5-digit chopping: 0.31415
5-digit rounding: 0.31416