DM 1: Write a small Matlab function `largest(a,b)` that returns the largest of the two values. Test that your function works by computing `largest(1,2)`, `largest(0,-1)` and `largest(5,5)`.

Solution:
Code:

```
function c = largest(a,b)
%The function largest(m,n) returns the larger value,
%if they are equal it returns the value.
if a<b
    c = b;
else
    c = a;
end
```

Output:

```
largest(1,2)

ans =

     2


largest(0,-1)

ans =

     0

largest(5,5)

ans =

     5
```

DM 2: Write a small Matlab function `nextprime(x)` that takes a positive integer argument and returns the smallest prime number at least as large as `x`. Your function should use a `while` loop and take advantage of the `isprime` function in Matlab. Test that your function works by computing `nextprime(5)`, `nextprime(6)`, `nextprime(-1)` and `nextprime(100)`.

Solution:
Code:

```
function y = nextprime(x)
% The function nextprime(x) outputs the least prime number
% that is greater than n.
y = x;
 if x<0
     y = 2;
     return
 end
while ~isprime(y)
    y = y+1;

end
```

Output:

```
nextprime(5)

ans =

     5


nextprime(6)

ans =

     7

nextprime(-1)

ans =

     2

nextprime(100)

ans =

   101
```

DM 3: Define a sequence of numbers by $x_1 = 1$ and $x_{k+1} = \frac{1}{2}x_k + 1$. Write a Matlab function `buildseq(N)` that returns an array with the first `N` elements of the sequence in it. For example, `buildseq(2)` should return $[1, 1.5]$. Test that your function works by computing the first four sequence elements by hand, and then verifying that your function computes them correctly. You may wish to take advantage of the Matlab command `zeros`.

Solution:
Code:

```
function X = buildseq(N)
% The buildseq(N) function returns an array with the first N values
% of the recurrence relation  x_i+1=(0.5*x_i) + 1.
x=0;
X = [];
for i=1:N
    x=(0.5*x) +1;
    X(i)= x;
end
end
```

Expected Output:

$$x_1 = 1$$
$$x_2 = \frac{1}{2} * (1) + 1 = \frac{3}{2}$$
$$x_3 = \frac{1}{2} * (\frac{3}{2}) + 1 = \frac{7}{4}$$
$$x_3 = \frac{1}{2} * (\frac{7}{4}) + 1 = \frac{15}{8}$$

Output:

```
buildseq(4)

ans =

   1.000000000000000    1.500000000000000    1.750000000000000
1.875000000000000
```

DM 4: Write a function `HW2bisect` that does the following:

1. Takes the following input:
   - `f`, the name of a function
   - Numbers `a` and `b` that (supposedly) bound an interval containing a root of `f`.
   - `delta`, a number determining the accuracy of the solution.

Your function should approximate a root of f in the interval $[a, b]$ by bisection. The approximation x should be within δ of a root The function should return:

1. `x`, the approximate root
2. A $n \times 2$ matrix `history` that contains a list of the interval endpoints at each stage, starting with the initial guess provided.

$$\texttt{hist} = \begin{pmatrix} a_0 & b_0 \\ a_1 & b_1 \\ \vdots & \vdots \\ a_n & b_n \end{pmatrix} \tag{1}$$

Your function should have help and should produce error messages appropriately when the function fails.

Test your function works by applying it to the function $f(x) = x^2 - 4$ with starting interval $[0, 5]$.

Code:

```
function [root,history] = HW2Bisect(a,b,f,d)
%initilizing history vectors
historya = [];
historyb = [];


fb = f(b);
fa = f(a);

%Checks if interval bounds contain root
if fa==0
    root = a;
    return
end


if fb==0
    root = b;
    return
end

% Syntax check for bisection algorithm
if sign(fa) == sign(fb)
    disp('Error: f(a) and f(b) have same sign.')
    return
end
```

```
% Bisection algorithm
count = 1;
while abs(a-b)> 2*d
    %iterate until interval width is <= 2xDelta
    c = (a+b)/2;
    fc = f(c);
    %assigning history values
    historya (count) = a;
    historyb (count) = b;
        if fc == 0
            root = c;
            break
        end
     if sign(fc) == sign(fa)
         a = c;
         fa = fc;
         count = count + 1;
         %assigning history values
         historya (count) = a;
         historyb (count) = b;
     else
         b = c;
         fb = fc;
         count = count + 1;
         %assigning history values
         historya (count) = a;
         historyb (count) = b;
     end
%disp(c);

 %Putting them together in one table
 history = [historya; historyb];
 %Flipping table for the correct format
 history = history';

end
root = (a+b)/2;

end
```

Output:

```
f = @(x) x^2 - 4
```

```
f =

    @(x)x^2-4

format long

[root, history] = HW2Bisect(0,5,f,1e-6)

root =

   1.999999880790710


history =

                   0   5.000000000000000
                   0   2.500000000000000
   1.250000000000000   2.500000000000000
   1.875000000000000   2.500000000000000
   1.875000000000000   2.187500000000000
   1.875000000000000   2.031250000000000
   1.953125000000000   2.031250000000000
   1.992187500000000   2.031250000000000
   1.992187500000000   2.011718750000000
   1.992187500000000   2.001953125000000
   1.997070312500000   2.001953125000000
   1.999511718750000   2.001953125000000
   1.999511718750000   2.000732421875000
   1.999511718750000   2.000122070312500
   1.999816894531250   2.000122070312500
   1.999969482421875   2.000122070312500
   1.999969482421875   2.000045776367188
   1.999969482421875   2.000007629394531
   1.999988555908203   2.000007629394531
   1.999998092651367   2.000007629394531
   1.999998092651367   2.000002861022949
   1.999998092651367   2.000000476837158
   1.999999284744263   2.000000476837158
```

Chapter 4: 2(a): How many steps would be required to reduce the size of this interval to 10^-12

Output:

```
f = @(x) (5 - x)*exp(x) - 5

f =

  <a href="matlab:helpPopup function_handle" style="font-weight:bold">fun

    @(x)(5-x)*exp(x)-5

[root, history] = HW2Bisect(4,5,f,1e-7)

root =

   4.965114176273346


history =

   4.000000000000000   5.000000000000000
   4.500000000000000   5.000000000000000
   4.750000000000000   5.000000000000000
   4.875000000000000   5.000000000000000
   4.937500000000000   5.000000000000000
   4.937500000000000   4.968750000000000
   4.953125000000000   4.968750000000000
   4.960937500000000   4.968750000000000
   4.964843750000000   4.968750000000000
   4.964843750000000   4.966796875000000
   4.964843750000000   4.965820312500000
   4.964843750000000   4.965332031250000
   4.965087890625000   4.965332031250000
   4.965087890625000   4.965209960937500
   4.965087890625000   4.965148925781250
   4.965087890625000   4.965118408203125
   4.965103149414063   4.965118408203125
   4.965110778808594   4.965118408203125
   4.965110778808594   4.965114593505859
   4.965112686157227   4.965114593505859
   4.965113639831543   4.965114593505859
   4.965114116668701   4.965114593505859
   4.965114116668701   4.965114355087280
   4.965114116668701   4.965114235877991
```

We know that gaining a digit of accuracy means decreasing the error (delta) by a factor of 10. Recall the convergence rate for the bisection algorithm is $\frac{1}{2}^n$ where n is the number of iterations through the algorithm. Thus we must solve the following equality,

$$\begin{aligned} \frac{1}{2}^n &= 10^-12 \\ n &= log_2(10^12) \\ &= 39.863 \\ &\geq 40 \end{aligned}$$

Thus at least 40 steps would be required to reduce the error to 10^{-12}

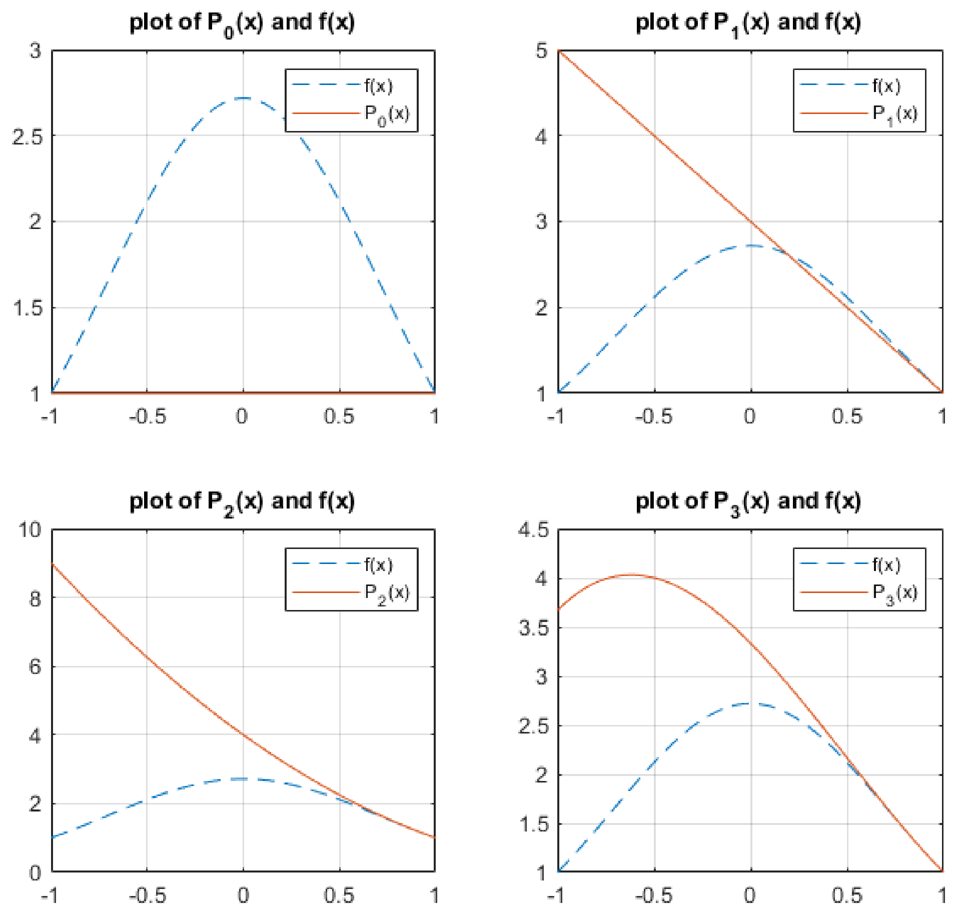
Chapter 4: 18: Produce a set of four plots of the first four Taylor polynomials of the function $f(x) = e^{1-x^2}$, expanded around the point $x = 1$.

Code:

```
% Plot the first four Taylor polynomials for exp(x).
z = 1;
x = [-1:.01:1];                    % Set x values
fx = exp(1 - x.^2);
p0 = ones(size(x));                % Compute Taylor polynomials
p1 = p0 + -2.*(x - z);
p2 = p1 + 1*(x - z).^2;
p3 = p2 + (2/3).*(x - z).^3;

subplot(2,2,1)                     % Plot results
plot(x,fx,'--', x,p0,'-');
legend('f(x)','P_0(x)')
title('plot of P_0(x) and f(x)')
grid
subplot(2,2,2)
grid
plot(x,fx,'--',x,p1,'-');
legend('f(x)','P_1(x)')
title('plot of P_1(x) and f(x)')
grid
subplot(2,2,3)
plot(x,fx,'--',x,p2,'-');
legend('f(x)','P_2(x)')
title('plot of P_2(x) and f(x)')
grid
subplot(2,2,4)
plot(x,fx,'--',x,p3,'-');
legend('f(x)','P_3(x)')
```

```
title('plot of P_3(x) and f(x)')
grid
```


plot of $P_0(x)$ and f(x)
plot of $P_1(x)$ and f(x)
plot of $P_2(x)$ and f(x)
plot of $P_3(x)$ and f(x)
f(x)
$P_0(x)$
$P_1(x)$
$P_2(x)$
$P_3(x)$