Problem 1
See attached code and output.

Problem 2 (B & F 2.2.19)

\[ x_n = \frac{1}{2} x_{n-1} + \frac{1}{x_{n-1}} \]
\[ = F(x_{n-1}) \]
\[ \text{with } F(x) = \frac{1}{2} x + \frac{1}{x} = \frac{x^2 + 2}{2x} \]

0) \[ \frac{d}{dx} F(x) = \frac{1}{2} - \frac{1}{x^2} \]

- \[ F \text{ maps } [\sqrt{2}, \infty) \text{ to } [\sqrt{2}, \infty) \text{ because:} \]

\[ F'(x) \geq 0 \text{ for } x \geq \sqrt{2} \Rightarrow F(x) \text{ is increasing} \]
\[ F'(\sqrt{2}) = 0 \Rightarrow F \text{ has a min at } x = \sqrt{2} \]
\[ \text{with } F(\sqrt{2}) = \sqrt{2} \]

- \[ F \text{ is a contraction on } [\sqrt{2}, \infty) \text{ because:} \]

\[ x > \sqrt{2} \Rightarrow 0 \leq \frac{1}{x^2} \leq \frac{1}{2} \Rightarrow 0 \leq F'(x) \leq \frac{1}{2} \leq 1 \]

\[ \Rightarrow \text{Thm 2.4 guarantees fixed point (i.e., } x_{n+1} = F(x_n) \text{ converges for all } x_0 > \sqrt{2} \text{ to } x^* = F(x^*), \text{ so } x^* = \sqrt{2}. \]
b) Let \( x_0 \in (0, \sqrt{2}) \) then:

\[
x_1 = F(x_0) = \frac{x_0^2 + 2}{2x_0}
\]

Since \( (x_0 - \sqrt{2})^2 = x_0^2 - 2\sqrt{2}x_0 + 2 > 0 \) we have

\[
x_1 = \frac{x_0^2 + 2}{2x_0} > \sqrt{2} \implies \text{method converges by part a).}
\]

c) We have shown in a) that method converges for \( x_0 > \sqrt{2} \) and

in b) for \( 0 < x_0 < \sqrt{2} \), thus method must converge

for any \( x_0 > 0 \).

d) Iteration can be obtained by applying Newton's method to

\[
f(x) = x^2 + 2
\]

Indeed:

\[
x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} = x_n - \frac{x_n^2 - 2}{2x_n} = x_n + \frac{1}{x_n}
\]

Problem 3 See attached code and output

Problem 4 \( x = \sqrt{\sqrt{\sqrt{\cdots}} = \sqrt{1 + \sqrt{1 + \sqrt{1 + \cdots}}}
\)

One can view \( x \) as the limit (if it exists) of the iteration:

\[
x_{n+1} = \sqrt{x_n + 1} = F(x_n)
\]
If this iteration has a limit then:

\[ x_0 = \sqrt{x_0 + p} \Rightarrow x_0^2 - x_0 - p = 0 \]

\[ \Rightarrow x_0 = \frac{1 \pm \sqrt{1 + 4p}}{2} \]

The only root that makes sense in this context is the positive one:

\[ x_0 = \frac{1 + \sqrt{1 + 4p}}{2} \]

(\( p = 1 \) we get golden section!)

Does this iteration have a limit? The easiest way to show this is by invoking the contractive mapping theorem.

\[ F(x) = \frac{1}{2\sqrt{p + x}} < \frac{1}{2} < 1 \text{ for } x + p > 1, x \geq 1 - p \]

Moreover, if \( x \in [1, \infty) \) we have:

- \( F'(x) \leq \frac{1}{2} < 1 \) (since \( p > 0 \))
- \( F(x) = \sqrt{p + x} > 1 \), thus \( F(x) \) maps \([1, \infty)\) onto itself.

\[ \Rightarrow \text{By contractive mapping theorem, iteration converges for any } x_0 > 1. \]
Problem 5 \[ F(x) = 4(1-x)x \]

- **F maps \([0, 1]\) to \([0, 1]\):**
  - Let \(x \in [0, 1]\) then:
    \[
    \begin{align*}
    x &> 0 \quad \Rightarrow \quad F(x) > 0 \\
    1-x &> 0 \quad \Rightarrow \quad F(x) < 0
    \end{align*}
    \]
  - Also \(F(x)\) has a max when \(F'(x) = 4(1-2x) = 0\)
    \[
    \Rightarrow x = \frac{1}{2} \quad \Rightarrow \quad F(x) \leq F(\frac{1}{2}) \leq 1
    \]
  - Thus \(\forall x \in [0, 1], \quad F(x) \in [0, 1]\).

- **F is not a contraction on \([0, 1]\)**
  - Indeed \(\forall x \in [0, \frac{1}{4}],\)
    \[
    \frac{4(1-\frac{1}{2})}{2} \leq F'(x) \leq \frac{4(1-0)}{4}
    \]
    \[
    \Rightarrow |F'(x)| \geq 2.
    \]

- **F has a fixed point:** \(\alpha = F(x)\) is a quadratic:
  \[
  4(1-x)x - x = 0
  \]
  \[
  \Rightarrow x(3-4x) = 0 \quad \Rightarrow \quad x = 0 \text{ or } x = \frac{3}{4}
  \]
  \(\alpha = \frac{3}{4}\) is a fixed point of \(F\).
This does not contradict FP theorem, because F is a contraction only a sufficient condition for having a fixed point.

That is, there could be functions not fixed points that are not contractions (such as this example).

Problem 6 (B & F 2.4.12)

We need to prove theorem 2.12:

The function \( f \in C^m [a, b] \) has a zero of multiplicity \( m \) if

\[ 0 = f(p) = f''(p) = \ldots = f^{(m-1)}(p) \]

but \( f^{(m)}(p) \neq 0 \).

**proof:**

\[ \Rightarrow \]

Assume \( f \) has a zero of multiplicity \( m \) at \( p \)

then:

\[ f(x) = (x-p)^m q(x), \text{ with } \lim_{x \to p} q(x) \neq 0 \]

Since \( f(x) \in C^m [a, b] \), we must have \( q \in C^m [a, b] \) as well.

\[ f'(p) = \lim_{x \to p} f'(x) \]
Write:
\[
 f^{(k)}(x) = \left( (x-p)^m q(x) \right)^{(k)} \\
 = \sum_{i=0}^{k} \binom{k}{i} (x-p)^m (x-p)^{(k-i)} q^{(k-i)}(x)
\]

Also:
\[
 (x-p)^m)^{(k)} = \begin{cases} 
 \frac{m!}{(m-k)!} (x-p)^{m-k}, & 0 \leq k \leq m \\
 0, & k > m
\end{cases}
\]

\[
\Rightarrow (x-p)^m)^{(k)}_{\alpha=p} = \begin{cases} 
 0 & \text{if } 0 \leq k \leq m-1 \\
 \frac{m!}{k!} & \text{if } k = m \\
 0 & \text{if } k > m+1
\end{cases}
\]

So \( f^{(k)}(p) = 0 \) for \( 0 \leq k \leq m-1 \)

but \( f^{(m)}(p) = m! q(p) \neq 0 \).

\[\lessgtr\]
Assume \( f^{(0)}(p) = f^{(1)}(p) = \ldots = f^{(m-2)}(p) = 0 \)

but \( f^{(m)}(p) \neq 0 \).

Using Taylor's theorem:
\[
 f(x) = \sum_{k=0}^{m-1} \frac{1}{k!} f^{(k)}(p) (x-p)^k + \frac{1}{m!} (x-p)^m f^{(m)}(\xi(x))
\]

\[
= 0 \text{ by Hyp.}
\]

Here \( \xi(x) \) is a number between \( x \) and \( p \).
Clearly $\xi(x) \rightarrow p$ as $x \rightarrow p$. Thus using that $f$ is continuous:

\[
\lim_{x \rightarrow p} f^{(m)}(\xi(x)) = f^{(m)}(\lim_{x \rightarrow p} \xi(x)) = f^{(m)}(p) \neq 0.
\]

We can thus write:

\[
f(x) = (x-p)^m q(x) \quad \text{where} \quad q(x) = \frac{1}{m!} f^{(m)}(\xi(x))
\]

and $\lim_{x \rightarrow p} q(x) \neq 0$. \(\square\)
% Bisection method to find root of function f
% in the interval [a,b] to within tol tolerance
%
% inputs:
%  f   function handle
%  a   left bound of interval
%  b   right bound of interval
%  delta desired tolerance for the root
%  eps  desired tolerance for f(root)
%  maxit maximum number of iterations
%
% outputs:
%  x      iterate history. The best approx found for the root is x(end)
%  a,b    interval containing the root
function [x,a,b] = bisection(f,a,b,delta,eps,maxit)

e = b-a; u = f(a); v = f(b);
% check if we can apply bisection method
if sign(u)*sign(v)>0
    fprintf('bisection: interval probably does not contain a root
');
    fprintf('bisection: please refine interval
');
    return;
end;

% prepare next iteration
for k = 1:maxit,
    e = e/2;
    x = [a c]; % save iterate
    w = f(c);
    if (abs(e)<delta || abs(w)<eps) % we are happy with this solution
        return;
    end;
    % prepare next iteration
    if (sign(w)*sign(u)<0)
        b=c; v=w; % root is in left half interval
    else
        a=c; u=w; % root is in right half interval
    end;
    c = a + e; % midpoint
    x = [x c]; % save iterate
    w = f(c);
end;

function xs = newton(f,fprime,x0,tol,maxit)
xs = [x0]; % store iterates
x = x0;
for k=1:maxit,
    xnew = x - f(x)/fprime(x);
    xs = [xs xnew]; % store iterates
    if abs(xnew-x)<tol break; end % check convergence
    x = xnew; % prepare next iteration
end;
Secant method to find a root of a function $f$ with initial guesses $x_0$ and $x_1$

Inputs
- $f$: function we want to find the root of
- $x_0$: initial iterate
- $x_1$: initial iterate
- $\text{tol}$: desired tolerance or precision
- $\text{maxit}$: maximum number of iterations

Outputs
- $\text{xs}$: iteration history, last iterate is $\text{xs}(\text{end})$

function $\text{xs} = \text{secant}(f,x_0,x_1,\text{tol},\text{maxit})$

$\text{xs} = [x_0 \ x_1]$; % store iterates
for $k=1:\text{maxit}$,
    $x_{\text{new}} = x_1 - f(x_1)*(x_1-x_0)/(f(x_1)-f(x_0))$;
    $\text{xs} = [\text{xs} \ x_{\text{new}}]$; % store iterates
if abs($x_{\text{new}}-x_1)<\text{tol}$; % check convergence
    $x_0=x_1$; $x_1=x_{\text{new}}$; % prepare next iteration
end;  

B&B 2.1.6 try out the bisection method in Matlab

$\text{tol} = 1e-5$; $\text{maxit}=30$

fprintf('***** problem B&B 2.1.6 c\n');

$\text{f} = @(x) x^2 - 4*4 + 4 - \log(x)$;

$x_1=\text{bisect}(f,1,2,\text{tol},\text{tol},\text{maxit})$;

fprintf('***** problem B&B 2.1.6 d\n');

$\text{f} = @(x) x + 1 - 2\sin(\pi x)$;

$x_2=\text{bisect}(f,0,0.5,\text{tol},\text{tol},\text{maxit})$;
% HW 2 Problem 3
	tol = 1e-5; maxit=20;
	fprintf('**** B&F 2.3.6 a and 2.3.8 a\n')

f = @(x) exp(x) + 2^(-x) + 2*cos(x) - 6;
fprime = @(x) exp(x) - log(2)*2^(-x) - 2*sin(x);
% compute root with Newton's and Secant methods
xnewt = newton(f,fprime,2,tol,maxit);
xsec  = secant(f,1,2,tol,maxit);
% display results
fprintf('%13s %13s
','newton','secant');
for i=1:max([length(xnewt),length(xsec)]),
    if (i>length(xnewt))
        fprintf('%13.8f %13s
',' ',xsec(i));
    else
        fprintf('%13.8f %13s
',xnewt(i),' '); 
    end;
end;

fprintf('**** B&F 2.3.6 b and 2.3.8 b
')

f = @(x) log(x-1) + cos(x-1);
fprime = @(x) 1/(x-1) + sin(x-1);
% compute root with Newton's and Secant methods
xnewt = newton(f,fprime,1.3,tol,maxit);
xsec  = secant(f,1.3,1.4,tol,maxit);
% display results
fprintf('%13s %13s
','newton','secant');
for i=1:max([length(xnewt),length(xsec)]),
    if (i>length(xnewt))
        fprintf('%13.8f %13s
',' ',xsec(i));
    else
        fprintf('%13.8f %13s
',xnewt(i),' '); 
    end;
end;