Homework 2.4: Tea time

pp. 75--, #1, 2, 4a-d, 5, 10, 11, 13, 22, 24, 26.

Exercise #1 and 2, p. 75

1. Write Octave code that implements Newton's method as a function.
2. Write Octave code that implements the secant method as a function.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Written by Dr. Len Brin          15 January 2012 %
% Purpose: Implementation of Newton's Method       %
% INPUT: function g and its derivative gp; initial %
%        value x0; tolerance TOL; maximum number   %
%        of iterations N0                          %
% OUTPUT: approximation x and number of iterations %
%        i; or message of failure                  %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [x,i] = newton(g,gp,x0,TOL,N0)
  i=1;
  while (i<=N0)
    x=x0-g(x0)/gp(x0);
    if (abs(x-x0)<TOL)
      return
    end%if
    i=i+1;
    x0=x;
  end%while
  x="Method failed---maximum number of iterations reached";
end%function
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Written by Dr. Len Brin       15 January 2014 %
% Purpose: Implementation of the Secant method. %
% INPUT: function g; initial values x0 and x1;  %
%        tolerance TOL; maximum iterations N0   %
% OUTPUT: approximation x and number of         %
%        iterations i; or message of failure    %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [x,i] = secant(g,x0,x1,TOL,N0)
  i=2;
  y0=g(x0);
  y1=g(x1);
  while (i<=N0)
    if (abs(y1-y0)<TOL)
      return
    end%if
    x=x1-y1*(x1-x0)/(y1-y0);
    if (abs(x-x1)<TOL)
      return
    end%if
    i=i+1;
    x0=x1;
    y0=y1;
    x1=x;
    y1=g(x1);
  end%while
  x="Method failed---maximum number of iterations reached";
end%function
```

Exercises 4a-d, p. 75

4. Use your Newton's method function from question 1 with a tolerance of 10^{-5} to find a solution of

   (a) $e^x + 2^{-x} + 2\cos x - 6 = 0$ using $1 \le x_0 \le 2$.

   (b) $\ln(x-1) + \cos(x-1) = 0$ using $1.3 \le x_0 \le 2$.

   (c) $2x\cos x - (x-2)^2 = 0$ using $2 \le x_0 \le 3$. [A]

   (d) $2x\cos x - (x-2)^2 = 0$ using $3 \le x_0 \le 4$. [A]

   (e) $(x-2)^2 - \ln x = 0$ using $1 \le x_0 \le 2$.

   (f) $(x-2)^2 - \ln x = 0$ using $e \le x_0 \le 4$.

4a

ans = 1.8294

4b

ans = 1.3977

4c

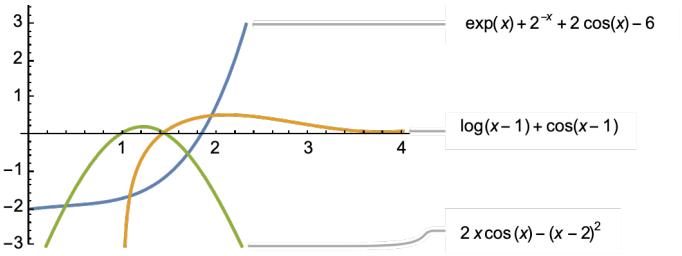
ans = 1.4788

4d

ans = 0.9484

In[138]:=
```
Plot[{Exp[x] + 2^(-x) + 2 * Cos[x] - 6,
  Log[x - 1] + Cos[x - 1], 2 * x * Cos[x] - (x - 2)^2}, {x, 0, 4},
 PlotLabels → Automatic, PlotRange → {-3, 3}]
```

Out[138]=



Exercise #5, pp. 75:

5. Repeat exercise 4 using your secant method code from question 2. Supply a value of x_1 from the same interval as x_0 but not equal to x_0. [A]

5a

ans = 1.8294

5b

ans = 1.3977

5c

ans = 1.4788

5d

ans = 0.9484

Exercise #10, pp. 75

10. Let $g(x) = 2\ln(1 + x^2) - x$. Find x_2 and x_3 using the secant method with

    (a) $x_0 = 5$ and $x_1 = 6$ [S]

    (b) $x_0 = 1$ and $x_1 = 2$

Exercise 10

10a

10.151

8.4346

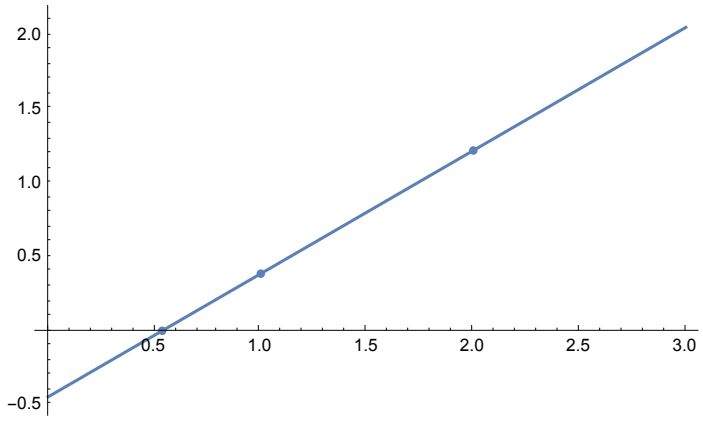
10a

0.5360

0.5722

In[183]:=
```
f[x_] := 2 Log[1 + x^2] - x
x0 = 1; y0 = f[x0];
x1 = 2; y1 = f[x1];
x2 = x1 - y1 * (x1 - x0) / (y1 - y0);
Show[
 Plot[y1 + (y1 - y0) / (x1 - x0) (x - x1), {x, 0, 3}],
 ListPlot[{{x0, y0}, {x1, y1}, {x2, 0}}, PlotStyle → {Large}]
]
```

Out[187]=



Exercise #11, pp. 75

11. Compare the secant method and Newton's method based on questions 5 and 4. Which finds roots in fewer iterations? Which one fails least often? Which is better?

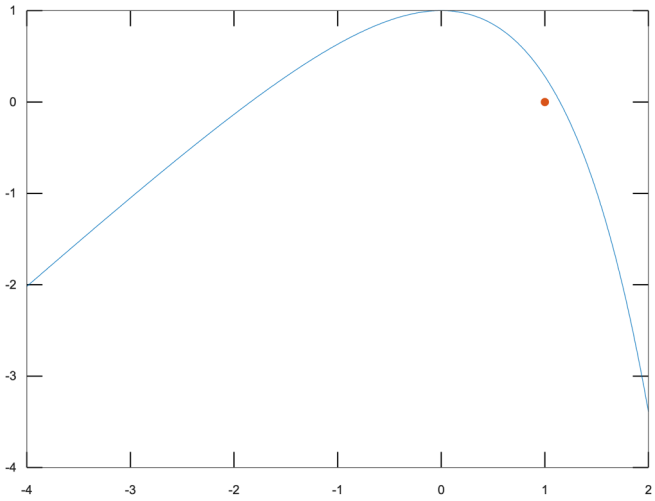
Given my initial values, they both found the same roots. Newton just found them faster:

Newton: # of iterations: 5,5,6,7

Secant: # of iterations: 7,8,8,13

Exercise #13, pp. 75

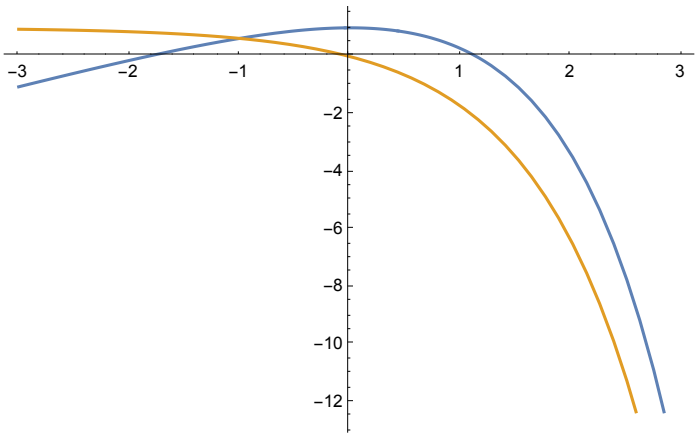
13. Find a value of x_0 for which Newton's method will fail to converge to a root of $g(x) = 2 + x - e^x$.



x=0 will cause Newton's to fail, because the derivative is 0 there.

In[190]:=
```
f[x_] := 2 + x - Exp[x]
Plot[{f[x], f'[x]}, {x, -3, 3}]
```

Out[191]=



Exercise #22, pp. 76

22. Suppose you are using the secant method with $x_0 = 1$ and $x_1 = 1.1$ to find a root of $f(x)$.

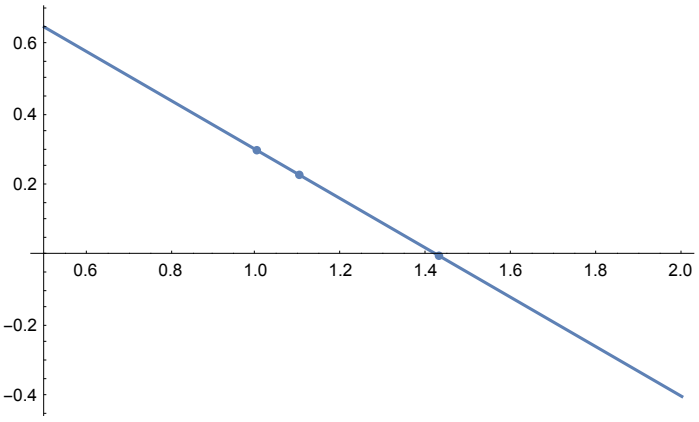
    (a) Find x_2 given that $f(1) = 0.3$ and $f(1.1) = 0.23$.

    (b) Create a sketch (graph) that illustrates the calculation. HINT: x_2 will be located where the line through $(x_0, f(x_0))$ and $(x_1, f(x_1))$ crosses the x-axis.

In[192]:=
```
x0 = 1; y0 = 0.3;
x1 = 1.1; y1 = 0.23;
x2 = x1 - y1 * (x1 - x0) / (y1 - y0)
Show[
 Plot[y1 + (y1 - y0) / (x1 - x0) (x - x1), {x, 0.5, 2}],
 ListPlot[{{x0, y0}, {x1, y1}, {x2, 0}}, PlotStyle → {Large}]
]
```

Out[194]= 1.42857

Out[195]=



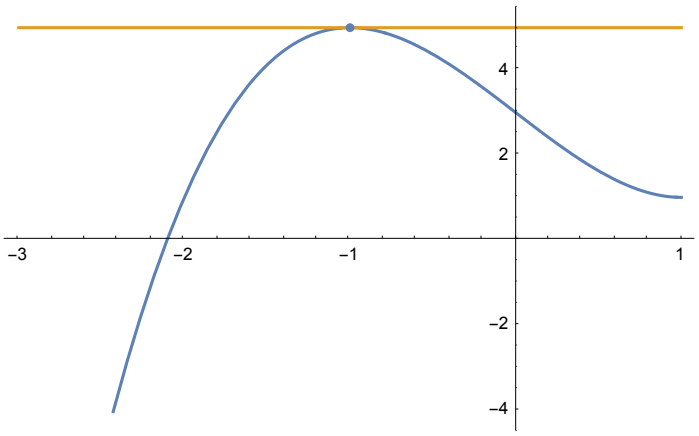
Exercise #24, pp. 76

24. Graph the polynomial $p(x) = x^3 - 3x + 3$, and demonstrate Newton's method graphically for $x_0 = -1$.

In[143]:=
```
p[x_] := x^3 - 3 x + 3;
x0 = -1; y0 = p[x0];
x1 = x0 - p[x0] / p'[x0];
Show[
 Plot[{p[x], y0 + p'[x0] (x - x0)}, {x, -3, 1}],
 ListPlot[{{x0, y0}, {x1, 0}}, PlotStyle → {Large}]
]
```

Power: Infinite expression $\frac{1}{0}$ encountered.

Out[146]=



Exercise #26, pp. 76

26. The sum of two numbers is 20. If each number is added to its square root, the product of the two sums is 172.2. Determine the two numbers to within 10^{-4} of their exact values. [S]

```
x+y=20
 (x+Sqrt[x])*(y+Sqrt[y])=172.2

 disp("Exercise 26");
f=@(x)(x + sqrt(x))*(20 - x + sqrt(20 - x)) - 172.2;
x0=9;
x1=9.3;
secant(f,x0,x1,10^(-4),20)
```

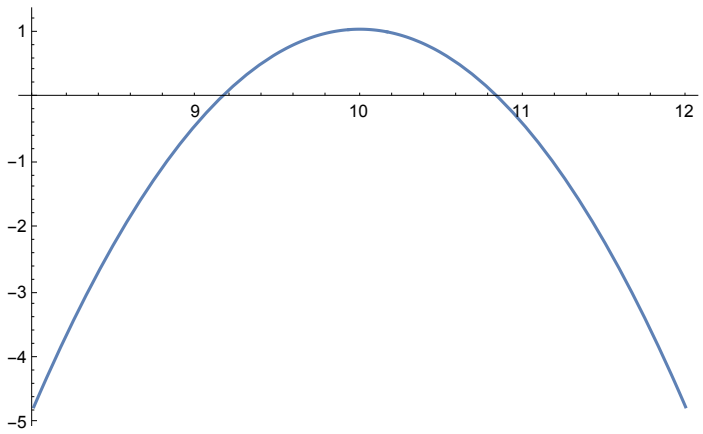
ans = 9.1496

In[147]:=
```
f[x_] := (x + Sqrt[x]) * (20 - x + Sqrt[20 - x]) - 172.2
f'[x]
Simplify[f[x]]
Plot[f[x], {x, 8, 12}]
Solve[n1 + n2 == 20 && (n1 + Sqrt[n1]) * (n2 + Sqrt[n2]) == 172.2, {n1, n2}]
```

Out[148]= $\left(1+\frac{1}{2\sqrt{x}}\right)\left(20+\sqrt{20-x}-x\right)+\left(-1-\frac{1}{2\sqrt{20-x}}\right)\left(\sqrt{x}+x\right)$

Out[149]= $-172.2+\left(20+\sqrt{20-x}-x\right)\left(\sqrt{x}+x\right)$

Out[150]=



Out[151]= $\{\{n1 \to 10.8504, n2 \to 9.14962\}, \{n1 \to 9.14962, n2 \to 10.8504\}\}$