CSC336 A4 Ramaneek Gill

1.
a)

CODE:
```
n = [0,1,2,3,4,5];
x=1;
x_of_i = x - sqrt(2);
fprintf(' n          x(n)                              x(n) - 
sqrt(2)\n~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\n');
for i = 0:5
        new_x = x-(x^2 - 2)/(2*x);
        x_of_i = x - sqrt(2);
        fprintf('%2d      %1.15e %1.15e\n',n(i+1), x, x_of_i);
        x = new_x;
end
```

OUTPUT:
```
 n     x(n)                  x(n) - sqrt(2)
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 0     1.000000000000000e+00   -4.142135623730951e-01
 1     1.500000000000000e+00   8.578643762690485e-02
 2     1.416666666666667e+00   2.453104293571595e-03
 3     1.414215686274510e+00   2.123901414741169e-06
 4     1.414213562374690e+00   1.594724352571575e-12
 5     1.414213562373095e+00   0.000000000000000e+00
```

b)
CODE:
```
f = @(x) x^2 - 2;
n = [0,1,2,3,4,5,6,7];
x(1) = 1;
x(2) = 2;
x_of_i(1) = x(1) - sqrt(2);
x_of_i(2) = x(2) - sqrt(2);

for j = 3:8
        x(j) = ((x(j-2)*f(x(j-1))) - (x(j-1)*f(x(j-2))))/(f(x(j-1)) - f(x(j-2)));
        x_of_i(j) = x(j) - sqrt(2);
end

fprintf(' n          x(n)                                      x(n) - 
sqrt(2)\n~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\n');

for i = 1:8
        fprintf('%2d      %1.15e %1.15e\n',n(i), x(i), x_of_i(i));
```

end

OUTPUT:

```
 n    x(n)                    x(n) - sqrt(2)
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 0    1.000000000000000e+00   -4.142135623730951e-01
 1    2.000000000000000e+00   5.857864376269049e-01
 2    1.333333333333333e+00   -8.088022903976189e-02
 3    1.400000000000000e+00   -1.421356237309523e-02
 4    1.414634146341463e+00   4.205839683681933e-04
 5    1.414211438474870e+00   -2.123898225070420e-06
 6    1.414213562057320e+00   -3.157747396898003e-10
 7    1.414213562373095e+00   0.000000000000000e+00
```

2.

a)

$$g1(x) = \frac{x^2+2}{3}$$
$$g1'(2) = \frac{2(2)}{3} = \frac{4}{3} > 1$$
$$g1'(2) > 1 => diverges$$

$$g2(x) = \sqrt{3x-2}, \qquad g2'(x) = \frac{3}{2\left(\sqrt{3x-2}\right)}$$
$$g2'(2) = \frac{3}{2\left(\sqrt{3(2)-2}\right)} = \frac{3}{4}$$
$$g2'(2) < 1 => converges\ linearly\ with\ a\ constant\ of\ 0.75$$

Type equation here.

b)

CODE:

```
g1 = @(x) (x^2+2)/3;
g2 = @(x) sqrt(3*x - 2);
g3 = @(x) 3-(2/x);
g4 = @(x) (x^2-2)/(2*x-3);

root = 2;
```

```
i = 0;
x = 2.336;
err = abs(x-root);

fprintf(' i          x                      err                            ratio\n');
fprintf('~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\n');

fprintf('|%3d | %20.12e | %20.12e |\n', i, x, err);

while i < 10
        x = g1(x);
        new_err = abs(x-root);
        ratio = new_err/err;
        err = new_err;

        fprintf('|%3d | %20.12e | %20.12e | %20.12e |\n', i, x, err, ratio);

        i = i + 1;

end
```

Note that to test g2, g3 and g4 all we need to do is change the
line x = g1(x) to x = g2(x) or x = g3(x) or x = g4(x)

OUTPUT for g1(x):

```
  i      x                         err                         ratio
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
|  0 |   2.336000000000e+00 |   3.360000000000e-01 |
|  1 |   2.485632000000e+00 |   4.856320000000e-01 |   1.445333333333e+00 |
|  2 |   2.726122146475e+00 |   7.261221464747e-01 |   1.495210666667e+00 |
|  3 |   3.143913985833e+00 |   1.143913985833e+00 |   1.575374048825e+00 |
|  4 |   3.961398383439e+00 |   1.961398383439e+00 |   1.714637995278e+00 |
|  5 |   5.897559050772e+00 |   3.897559050772e+00 |   1.987132794480e+00 |
|  6 |   1.226040091911e+01 |   1.026040091911e+01 |   2.632519683591e+00 |
|  7 |   5.077247689913e+01 |   4.877247689913e+01 |   4.753466973038e+00 |
|  8 |   8.599481368242e+02 |   8.579481368242e+02 |   1.759082563304e+01 |
|  9 |   2.465042660091e+05 |   2.465022660091e+05 |   2.873160456081e+02 |
|  10 |   2.025478438756e+10 |   2.025478438556e+10 |   8.216875533637e+04 |
```

OUTPUT for g2(x):

```
  i      x                         err                         ratio
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
|  0 |   2.336000000000e+00 |   3.360000000000e-01 |
|  1 |   2.237856116912e+00 |   2.378561169119e-01 |   7.079051098568e-01 |
|  2 |   2.171075390385e+00 |   1.710753903851e-01 |   7.192389777743e-01 |
|  3 |   2.124435494703e+00 |   1.244354947033e-01 |   7.273722679995e-01 |
```

4	2.091245199423e+00	9.124519942303e-02	7.332730877199e-01
5	2.067301525726e+00	6.730152572601e-02	7.375897707669e-01
6	2.049854769777e+00	4.985476977713e-02	7.407673041483e-01
7	2.037047939871e+00	3.704793987068e-02	7.431172591168e-01
8	2.027595575950e+00	2.759557595001e-02	7.448612809871e-01
9	2.020590687856e+00	2.059068785591e-02	7.461590181416e-01
10	2.015383850180e+00	1.538385018034e-02	7.471265791601e-01

OUTPUT for g3(x):

i	x	err	ratio
0	2.336000000000e+00	3.360000000000e-01	
1	2.143835616438e+00	1.438356164384e-01	4.280821917808e-01
2	2.067092651757e+00	6.709265175719e-02	4.664536741214e-01
3	2.032457496136e+00	3.245749613601e-02	4.837712519320e-01
4	2.015969581749e+00	1.596958174905e-02	4.920152091255e-01
5	2.007921539042e+00	7.921539041871e-03	4.960392304791e-01
6	2.003945143716e+00	3.945143715950e-03	4.980274281420e-01
7	2.001968688478e+00	1.968688478485e-03	4.990156557608e-01
8	2.000983376258e+00	9.833762584877e-04	4.995083118708e-01
9	2.000491446491e+00	4.914464908383e-04	4.997542767548e-01
10	2.000245662880e+00	2.456628803387e-04	4.998771685593e-01

OUTPUT for g4(x):

i	x	err	ratio
0	2.336000000000e+00	3.360000000000e-01	
1	2.067521531100e+00	6.752153110048e-02	2.009569377990e-01
2	2.004016726161e+00	4.016726161308e-03	5.948807877787e-02
3	2.000016005510e+00	1.600550955727e-05	3.984715142259e-03
4	2.000000000256e+00	2.561684198099e-10	1.600501495396e-05
5	2.000000000000e+00	0.000000000000e+00	0.000000000000e+00
6	2.000000000000e+00	0.000000000000e+00	NaN
7	2.000000000000e+00	0.000000000000e+00	NaN
8	2.000000000000e+00	0.000000000000e+00	NaN
9	2.000000000000e+00	0.000000000000e+00	NaN
10	2.000000000000e+00	0.000000000000e+00	NaN

The result for this becomes NaN because of dividing by 0.
g4 converges quickly since it was shown in (a) it is quadratic.
g3 converges slowly at approximately a rate of 0.5 when you look at the err column in each iteration.
g2 converges exactly like g3 except at a rate of 0.75.
g1 is producing wild results and doesn’t look like it will converge, as it is implied in part a’s calculations.

3.
CODE:

```
R = 0.082054;
```

```
b = 0.04267;
a = 3.592;
K = 300;

fprintf('p   v(gas law) v(van der Walls) \n');
fprintf('~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\n');
for i = 1:3
        pressure = 10^(i-1);
        f = @(v) (pressure + a/v^2) * (v - b) - R * K;
        v_law = R * K/pressure;
        v_van = fzero(f, v_law);
        fprintf('|%4.0f | %10.7f | %10.12f | \n', pressure, v_law, v_van);
end
```

OUTPUT:

```
p       v(gas law)      v(van der Walls)
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
|   1 | 24.6162000 | 24.512588128442 |
|  10 |  2.4616200 | 2.354495580702 |
| 100 |  0.2461620 | 0.079510827813 |
```

4.

a)

given: f(x) = 1/x –b

then: $f'(x) = -1/x^2$

Newton's Method: $x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$

Calculations:

$$x_{n+1} = x_n - \frac{\frac{1}{x_n} - b}{-\frac{1}{x_n^2}}$$
$$x_{n+1} = x_n - \left(\frac{1}{x_n} - b\right)(-x_n^2)$$
$$x_{n+1} = x_n + x_n - b(x_n^2)$$
$$x_{n+1} = x_n(2 - bx_n)$$

As you can see now this equation doesn't use division, it is also computationally efficient because only 2 multiplications and one subtraction need to be computed and they will also never catastrophically cancel.

b)

$$x_{n+1} = x_n(2 - bx_n)$$

The error for this computation is:

$$\delta_1(\delta_2 + \delta_3)$$

Where δ_1 is the error of multiplying x_n with the result in the brackes, δ_2 is the error of the subtraction and δ_3 is the error of the multiplication of $b(x_n)$.

For r0 the error will be $\delta \leq \frac{1}{2}\epsilon_{mach}$. After we run 1 iteration we will compute r1 which is based on r0's result. This will make $\delta^2 \leq \frac{1}{4}\epsilon_{mach}$.

Therefore the error is satisfied for r1 being the square of r2's computation error.

c)
Since the error for r1 is approximately to the square of r0 it implies that r1 will have twice as many correct digits than r0 if it is less than 1.