1. The Gaussian Quadrature rules are defined to minimize the expected error obtained in the approximation formula given by $\int_a^b f(x)\ dx = \sum_{i=1}^{n} c_i * f(x_i)$ by choosing the weights (c_i) and nodes (x_i) that give the greatest degree of precision. The various approaches for defining the weight and nodes of the quadrature formulas to obtain the desired precision and how they are different from the Newton-Cotes formulas will be discussed.

One approach for solving Gaussian Quadrature formulas is to use brute force by solving systems of nonlinear equations [1]. This approach can be done by formulating a series of equations, putting them in matrix form, and solving the matrix for a given (n) and precision. The amount of nonlinear equations is determined by the precision or known as the degrees. The equations that are generated are in the form $\sum_{i=1}^{n} c_i * f(x_i)$ and are made equal to the integral of x^idx for (i) from 0 to the amount of degrees that was determined. These equations are then solved as a matrix and the (x) nodes and (c) weights are obtained. Another approach is to use Legendre polynomials and a theorem that says $c_i = \int_{-1}^{1} \prod_{\substack{j=1 \\ j\neq 1}}^{n} \frac{x-x_j}{x_1-x_j}\, dx$ [2]. Every Legendre polynomial $P_n(x)$ has its own equation. For example, $P_2(x) = x^2 - \frac{1}{3}$. Using this, (x) can be solved by setting the equation to zero which will result in (n) number of (x) nodes. Then, these x nodes can be substituted in the c_i equation and will give each corresponding (c) or weight values.

Gaussian Quadrature formulas are vastly different from Newton-Cotes rules. Gaussian Quadrature rules pick the best points in the interval of approximation whereas Newton Cotes rules pick equally spaced points. So, Gaussian rules avoid the restriction that Newton Cotes rules have and can be computed with unequally spaced nodes. Another difference is that Gauss rules try to maximize precision with 2n-1 using the optimal node locations while Newton Cotes have a fixed precision. Because of this, Gauss Quadrature rules are more accurate than Newton Cotes rules.

[1] G. H. Golub and G. Meurant (2010) Matrices, Moments and Quadrature with Applications. Princeton University Press: Princeton, NJ.

[2] R. L. Burden, D. J. Faires, and A. M. Burden (2016) Numerical Analysis. Cengage Learning.

2a)

a) Since we want a precision of 2n-1 and we are given n=3, we want to form equations up to degree 5 since 2*3-1 = 5.

Degree 0:

$$\int_{-1}^{1} 1 \; dx = x| \begin{matrix}1\\-1\end{matrix} = 1-(-1) = 2$$

$$\int_{-1}^{1} 1 \, dx => 2 = c1 * f(x1) + c2 * f(x2) + c3 * f(x3)$$

$$c_1 + c_2 + c_3 = 2$$

Degree 1:

$$\int_{-1}^{1} x \; dx = \frac{x^2}{2}| \begin{matrix}1\\-1\end{matrix} = \frac{1}{2} - \frac{1}{2} = 0$$

$$c_1 * x_1 + c_2 * x_2 + c_3 * x_3 = 0$$

Degree 2:

$$\int_{-1}^{1} x^2 \; dx = \frac{x^3}{3}| \begin{matrix}1\\-1\end{matrix} = \frac{1}{3} - (-\frac{1}{3}) = \frac{2}{3}$$

$$c_1 * {x_1}^2 + c_2 * {x_2}^2 + c_3 * {x_3}^2 = \frac{2}{3}$$

Degree 3:

$$\int_{-1}^{1} x^3 \, dx = \frac{x^4}{4}| \begin{matrix}1\\-1\end{matrix} = \frac{1}{4} - \frac{1}{4} = 0$$

$$c_1 * {x_1}^3 + c_2 * {x_2}^3 + c_3 * {x_3}^3 = 0$$

Degree 4:

$$\int_{-1}^{1} x^4 \; dx = \frac{x^5}{5}| \begin{matrix}1\\-1\end{matrix} = \frac{1}{5} - (-\frac{1}{5}) = \frac{2}{5}$$

$$c_1 * {x_1}^4 + c_2 * {x_2}^4 + c_3 * {x_3}^4 = \frac{2}{5}$$

Degree 5:

$$\int_{-1}^{1} x^5 \; dx = \frac{x^6}{6}| \begin{matrix}1\\-1\end{matrix} = \frac{1}{6} - \frac{1}{6} = 0$$

$$c_1 * {x_1}^5 + c_2 * {x_2}^5 + c_3 * {x_3}^5 = 0$$

Plug equations into the vector F(x) =0 where x is the vector x.

$$F = @(x)\begin{bmatrix} c_1 + c_2 + c_3 - 2 \\ c_1 * x_1 + c_2 * x_2 + c_3 * x_3 - 0 \\ c_1 * {x_1}^2 + c_2 * {x_2}^2 + c_3 * {x_3}^2 - \frac{2}{3} \\ c_1 * {x_1}^3 + c_2 * {x_2}^3 + c_3 * {x_3}^3 - 0 \\ c_1 * {x_1}^4 + c_2 * {x_2}^4 + c_3 * {x_3}^4 - \frac{2}{5} \\ c_1 * {x_1}^5 + c_2 * {x_2}^5 + c_3 * {x_3}^5 - 0 \end{bmatrix}$$

Plug the above matrix in MATLAB and use fsolve to find the solution. Please see my file named Project_Problem2A.m to see how I did this.
Note: In MATLAB,
x1, x2, x3 are c1, c2, c3 respectively
x3, x4, x5 are x1, x2, x3 respectively

The result are the weights (c_i) and nodes (x_i) for n=3:

c_1 = 0.555555555443042
c_2 =0.888888889113917
c_3 =0.555555555443042
x_1 =0.774596669368293
x_2 = 0.000000000000000
x_3 =-0.774596669368293

b) $P_3(x) = x^3 - \frac{3}{5}x$

$x^3 - \frac{3}{5}x = 0$ Solve for x.

$x(x^2 - \frac{3}{5}) = 0$

$x_1 = -\sqrt{\frac{3}{5}} = -\frac{\sqrt{15}}{5}$ $\quad x_2 = 0 \quad$ $x_3 = \sqrt{\frac{3}{5}} = \frac{\sqrt{15}}{5}$

$$c_1 = \int_{-1}^{1} \prod_{\substack{j=1 \\ j\neq 1}}^{3} \frac{x - x_j}{x_1 - x_j} dx = \int_{-1}^{1} \frac{x - x_2}{x_1 - x_2} * \frac{x - x_3}{x_1 - x_3} = \int_{-1}^{1} \frac{x - 0}{-\frac{\sqrt{15}}{5} - 0} * \frac{x - \frac{\sqrt{15}}{5}}{-\frac{\sqrt{15}}{5} - \frac{\sqrt{15}}{5}}$$

$$c_2 = \int_{-1}^{1} \frac{x - (-\frac{\sqrt{15}}{5})}{0 - (-\frac{\sqrt{15}}{5})} * \frac{x - \frac{\sqrt{15}}{5}}{0 - \frac{\sqrt{15}}{5}} \qquad c_3 = \int_{-1}^{1} \frac{x - (-\frac{\sqrt{15}}{5})}{\frac{\sqrt{15}}{5} - (-\frac{\sqrt{15}}{5})} * \frac{x - 0}{\frac{\sqrt{15}}{5} - 0}$$

Plug these equation in MATLAB to solve. Please see my file: Project_Problem2A.m
The results are the same weights and nodes obtained from part (a) of this problem:
x_1 =

0.774596669241483

c_1 =

  0.555555555555556

x_2 =

   0

c_2 =

  0.888888888888889

x_3 =

  -0.774596669241483

c_3 =

  0.555555555555556

2b) Please see file named GaussTable.m for MATLAB code for generating the following results:

```
>> [nodes, weights] = GaussTable(2)
Table for n=2

ans =

  2×2 table

      nodes           weights
    _____________    _______
    0.5773502692      1
   -0.5773502692      1

>> [nodes, weights] = GaussTable(3)
Table for n=3

ans =

  3×2 table

       nodes              weights
    ______________    _____________

    0.7745966692      0.5555555556
          0           0.8888888889
   -0.7745966692      0.5555555556

>> [nodes, weights] = GaussTable(4)
Table for n=4

ans =

  4×2 table

       nodes             weights
    ______________    _____________

    0.8611363116      0.3478548451
    0.3399810436      0.6521451549
   -0.3399810436      0.6521451549
   -0.8611363116      0.3478548451

>> [nodes, weights] = GaussTable(5)
Table for n=5

ans =

  5×2 table

       nodes              weights
    ______________    _____________

    0.9061798459     0.236926885
```

0.5384693101	0.4786286705
0	0.5688888889
-0.5384693101	0.4786286705
-0.9061798459	0.236926885

2c) Please see file named GaussApprox.m for MATLAB code for generating the following results:

```
GLApprox2 =

  0.636196564962864


abssoluteError2 =

    1.678080636036139e-05


GLApprox3 =

  0.636213195944744


abssoluteError3 =

    1.498244804887250e-07


GLApprox4 =

  0.636213344399012


abssoluteError4 =

    1.370212276974314e-09

GLApprox5 =

  0.636213345724370

abssoluteError5 =

    4.485434246248587e-11
```