```
% Defining the function and its derivative
f = @(x) 6 - 3.*x.*(1 + exp(3.*(1-x)));
df = @(x) -3*(1 + exp(3*(1-x))) + 9*x*exp(3*(1-x));

% a. Use the MATLAB built-in function fzero with initial guess 0.1 to
 determine the value of x1.
A1 = fzero(f, 0.1);

% b. Use the MATLAB built-in function fzero with initial guess 1.9 to
 determine the value of x2.
A2 = fzero(f, 1.9);

% c. Bisection Method for 0 # x # 0.5
[A3, A4] = bisectionMethod(f, [0, 0.5], 10^-15);

% d. Bisection Method for 1.5 # x # 2
[A5, A6] = bisectionMethod(f, [1.5, 2], 10^-15);

% e. Newton's Method for x0 = 0.1
[A7, A8] = newtonMethod(f, df, 0.1, 10^-15);

% f. Newton's Method for x0 = 1.9
[A9, A10] = newtonMethod(f, df, 1.9, 10^-15);

% g. Secant Method for x0 = 0.1, x1 = 0.11
[A11, A12] = secantMethod(f, 0.1, 0.11, 10^-15);

% h. Secant Method for x0 = 1.9, x1 = 1.8
[A13, A14] = secantMethod(f, 1.9, 1.8, 10^-15);

% Defining the bisectionMethod function
function [x, N] = bisectionMethod(f, int, thresh)
    x = [int(1) mean(int) int(2)];
    N = 0;
    while x(3)-x(1) >= thresh
        fx = f(x);
        if fx(1)*fx(2) < 0
            x = [x(1) mean([x(1) x(2)]) x(2)];
        elseif fx(2)*fx(3) < 0
            x = [x(2) mean([x(2) x(3)]) x(3)];
        else
            x = [x(2) x(2) x(2)];
        end
        N = N +1;
    end
    x = x(2);
end

% Defining the newtonMethod function
function [x1, N] = newtonMethod(f, df, x0, thresh)
    x1 = x0 - f(x0)/df(x0);
    err = abs(x1-x0);
```

```
    N = 1;
    while err >= thresh
        x0 = x1;
        x1 = x0 - f(x0)/df(x0);
        err = abs(x1-x0);
        N = N+1;
    end
end

% Secant Method function
function [x2, N] = secantMethod(f, x0, x1, thresh)
    err = abs(x1-x0);
    N = 0;  % Initialize N to 0
    while err >= thresh
        x2 = x1 - (f(x1)*(x1-x0))/(f(x1)-f(x0)); % Update rule for Secant
 Method
        err = abs(x2-x1);
        x0 = x1;
        x1 = x2;
        N = N+1;
    end
end
```

Published with MATLAB® R2022b