```matlab
% Computes the root of a function using the bisection method.
% Written by Zachary Ferguson


% Solve the project questions.
function bisection_method()
    fprintf('Bisection Method\nWrtten by Zachary Ferguson\n\n')

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Q1a
    fprintf('Q1a:\n\tf(x) = x^3 - 9\n');
    f = @(x) x^3 - 9;
    r = compute_root(f, 2, 3);
    fprintf('\tr = %.10f\n', r);
    y = 3^(2. / 3.);
    print_errors(f, r, y);

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Q1b
    fprintf('Q1b:\n\tf(x) = 3x^3 + x^2 - x - 5\n');
    f = @(x) 3 * x^3 + x^2 - x - 5;
    r = compute_root(f, 1, 2);
    fprintf('\tr = %.10f\n', r);
    y = (1. / 9. * (-1 + (593 - 27 * (481^0.5))^(1.0 / 3.0) +...
        (593 + 27 * (481^0.5))^(1.0 / 3.0)));
    print_errors(f, r, y);

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Q1c
    fprintf('Q1c:\n\tf(x) = cos^2(x) - x + 6\n');
    f = @(x) cos(x) * cos(x) - x + 6;
    r = compute_root(f, 6, 7);
    fprintf('\tr = %.10f\n', r);
    y = 6.7760923163195023262;
    print_errors(f, r, y);

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Q3a
    fprintf('Q3a:\n\tf(x) = 2x^3 - 6x - 1\n');
    f = @(x) 2.0 * x^3 - 6 * x - 1;

    r = compute_root(f, -2, -1);
    fprintf('\tr1 = %.10f\n', r);
    y = -1.64178352745293;
    print_errors(f, r, y);

    r = compute_root(f, -1, 0);
    fprintf('\tr2 = %.10f\n', r);
    y = -0.168254401781027;
    print_errors(f, r, y);

    r = compute_root(f, 1, 2);
    fprintf('\tr3 = %.10f\n', r);
    y = 1.81003792923395;
    print_errors(f, r, y);

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Q3b
    fprintf('Q3b:\n\tf(x) = e^(x-2) + x^3 - x\n');
    f = @(x) exp(x - 2) + x^3 - x;

    r = compute_root(f, -2, -1);
    fprintf('\tr1 = %.10f\n', r);
    y = -1.0234821948582364944;
    print_errors(f, r, y);
```

```
    r = compute_root(f, -0.5, 0.5);
    fprintf('\tr2 = %.10f\n', r);
    y = 0.16382224325010849634;
    print_errors(f, r, y);

    r = compute_root(f, 0.5, 1.5);
    fprintf('\tr3 = %.10f\n', r);
    y = 0.78894138905554556637;
    print_errors(f, r, y);

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Q3c
    fprintf('Q3c:\n\tf(x) = 1 + 5x - 6x^3 - e^(2x)\n');
    f = @(x) 1 + 5 * x - 6 * x^3 - exp(2 * x);

    r = compute_root(f, -1.5, -0.5);
    fprintf('\tr1 = %.10f\n', r);
    y = -0.81809373448119542124;
    print_errors(f, r, y);

    r = compute_root(f, -0.6, 0.4);
    fprintf('\tr2 = %.10f\n', r);
    y = 0.0;
    print_errors(f, r, y);

    r = compute_root(f, 0.5, 1.5);
    fprintf('\tr3 = %.10f\n', r);
    y = 0.50630828634622119599;
    print_errors(f, r, y);

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Q4a
    fprintf('Q4a:\n\tf(x) = x^2 - A\n');
    fprintf('\tA = 2, (a, b) = (1, 2)\n');
    A = 2;
    f = @(x) x^2 - A;
    r = compute_root(f, 1, 2);
    fprintf('\tr = %.10f\n', r);
    y = 2^0.5;
    print_errors(f, r, y);

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Q4c
    fprintf('Q4b:\n\tA = 3, (a, b) = (1, 2)\n');
    A = 3;
    f = @(x) x^2 - A;
    r = compute_root(f, 1, 2);
    fprintf('\tr = %.10f\n', r);
    y = 3^0.5;
    print_errors(f, r, y);

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Q4c
    fprintf('Q4c:\n\tA = 5, (a, b) = (2, 3)\n');
    A = 5;
    f = @(x) x^2 - A;
    r = compute_root(f, 2, 3);
    fprintf('\tr = %.10f\n', r);
    y = 5^0.5;
    print_errors(f, r, y);
end


% Finds the root of the function, f, in the interval [a, b] within an
```

```
% absolute tolerance.
function r = compute_root(f, a, b, tol)
    if nargin < 4
        tol = 1e-7;
    end
    assert(f(a) * f(b) <= 0);

    n = 0;
    while (abs(b - a) / 2.0) > (0.5 * tol)
        c = (a + b) / 2.0;
        if (f(c) == 0)
            break;
        end

        if (f(a) * f(c) <= 0)
            b = c;
        else
            a = c;
        end
        n = n + 1;
    end
    fprintf('\tn = %d\n', n);
    r = (a + b) / 2.0;
end


% Print the forward and backward error of r.
function print_errors(f, r, y)
    fprintf('\tForward error: %.10f\n', abs(y - r))
    fprintf('\tBackward error: %.10f\n', abs(f(r)))
end
```