Computer Exercise 3.2.2

This program will solve the equation $x^3 + 2x^2 + 10x = 20$ by finding the root of $f(x) = x^3 + 2x^2 + 10x - 20$ closest to an initial point $x_0 = 2$. Moreover, $f(x)$ will be casted via nested multiplication as $f(x) = x(10 + x(2 + x(1))) - 20$ for efficient computation of the polynomial.

The program will terminate when either the number of iterates exceeds 10 or when the following error condition is met: $|x_n - x_{n-1}| < \frac{1}{2} \times 10^{-5}$

Iterate number, corresponding x value, and the the function at x will be displayed.

```
syms x; %program is modified to only take a function
        %as input by having the derivative evaluated
        %within the Newton algorithm; to accomplish this
        %symbolic functions are utilized

%%inputs for Newton's method

f = x*(10 + x*(2 + x*(1))) - 20; %symbolic function

x0=2; %initial point

N=10; %max number of iterates

err = 0.5 * 10^(-5); %error tolerance for successive points

m=1; %if root multiplicity is known ahead of time,
     %this value can be changed to account for
     %root multiplicity and use the modified
     %Newton's method

root1 = newton(f, x0, N, err, m);
```

```
n = 0, xn =   2.000000000000, f(xn) = 1.600000000000e+01, error = 1.000000000000e+00
n = 1, xn =   1.466666666667, f(xn) = 2.123851851852e+00, error = 5.333333333333e-01
n = 2, xn =   1.371512013806, f(xn) = 5.708664190432e-02, error = 9.515465286075e-02
n = 3, xn =   1.368810222634, f(xn) = 4.461440696415e-05, error = 2.701791172025e-03
n = 4, xn =   1.368808107823, f(xn) = 2.731055306559e-11, error = 2.114811228029e-06
```

The program terminates after four iterates have been evaluated. The error has also been displayed; the quadratic convergence of Newton's method becomes evident from the third to fourth iterations. We can see that the root x_r converging to four decimal places so we can conclude that $x_r \approx 1.3688$.

```
function root = newton(f, x0, N, err, m)
    n=0; %initialize iterate
    syms x; %symbols needs to be redeclared otherwise
            %algorithm yields an error

    fd = diff(f); %symbolically evaluate derivative of
                  %input function
    y = subs(f, x, x0); %evaluate input function at x0

    dy = subs(fd, x, x0); %evaluate derivative at x0

    xn = x0; %initialize iterate x value for loop

    error = 1; %initialize error to any value such that error < err

    fprintf(['n = %d, xn = %16.12f, f(xn) = %16.12e, ' ...
        'error = %16.12e \n'], n, xn, y, error)
    %display the zeroth iterate
    while (error > err) && (n < N)
        root = xn - ((m*y)/dy); %evaluate subsequent point
                                %using Newton's method formula
        y = subs(f, x, root);
        dy = subs(fd, x, root);
        error = abs(xn -root); %error value between successive points
        xn = root;
        n = n + 1; %increment iteration number
        fprintf(['n = %d, xn = %16.12f, f(xn) = %16.12e, ' ...
            'error = %16.12e \n'], n, xn, y, error)
        %display nth iterate
    end
end
```