```
%PROGRAM CODE CREDIT: Original version found on page 171-172
%   of Numerical Analysis, Second Edition by Timothy Sauer

%Program 3.6 Cubic spline plot
%Computes and plots spline from data points
%Input: x, y vectors of data points, number k of plotted points per segment
%Output: x1, y1 spline values at plotted points

function [ x1, y1 ] = CSCI3656_ProblemSet7_3( x, y, k )
n = length(x);
coefficient = splinecoeff(x, y);
x1 = [];
y1 = [];

for i = 1: n - 1
    xs = linspace(x(i), x(i + 1), k + 1);
    dx = xs - x(i);
    % Evaluate using nested multiplication
    ys = coefficient(i, 3) * dx;
    ys = (ys + coefficient(i, 2)).*dx;
    ys = (ys + coefficient(i, 1)).*dx + y(i);
    x1 = [x1; xs(i: k)'];
    y1 = [y1; ys(i: k)'];
end

x1 = [x1; x(end)];
y1 = [y1; y(end)];
plot(x, y, 'o', x1, y1)
plottools

%Program 3.5 Calculation of spline coefficients
%Calculates coefficients of cubic spline
%Input: x, y vectors of data points plus two optional extra data v1, vn
%Output: matrix of coefficients b1, c1, d1; b2, c2, d2;...
function coefficent = splinecoeff( x, y )
n = length(x);
v1 = 0;
vn = 0;
% Matrix A is nxn
matrixA = zeros(n, n);
r = zeros(n, 1);

% Define the deltas
for i = 1: n - 1
    deltaX(i) = x(i + 1) - x(i); dy(i) = y(i + 1) - y(i);
end

% Load the A matrix
```

```
for i = 2: n - 1
    matrixA(i, i - 1: i + 1) = [deltaX(i - 1) 2*(deltaX(i - 1) + deltaX(i))
deltaX(i)];
    % Right-hand side
    r(i) = 3 * (dy(i) / deltaX(i) - dy(i-1) / deltaX(i-1));
end

% Set endpoint conditions
% Use only one of following 5 pairs:
%  Natural spline conditions
matrixA(1, 1) = 1;
matrixA(n, n) = 1;

%  Curvature-adj conditions
%matrixA(1, 1) = 2;
%r(1) = v1;
%matrixA(n, n) = 2;
%r(n) = vn;

% Clamped
%matrixA(1, 1: 2) = [2*deltaX(1) deltaX(1)];
%r(1) = 3 * (dy(1) / deltaX(1) - v1);
%matrixA(n, n - 1: n) = [deltaX(n - 1) 2*deltaX(n - 1)];
%r(n) = 3 * (vn - dy(n - 1) / deltaX(n - 1));

%  Parabol-term conditions, for n >= 3
%matrixA(1, 1: 2) = [1 -1];
%matrixA(n, n - 1: n) = [1 -1];

%  Not-a-knot, for n >= 4
%matrixA(1, 1: 3) = [deltaX(2) -(deltaX(1) + deltaX(2)) deltaX(1)];
%matrixA(n, n - 2: n) = [deltaX(n - 1) -(deltaX(n - 2) + deltaX(n - 1)) deltaX(n -
2)];

coefficent = zeros(n, 3);
% Solves for c coefficients
coefficent(:, 2) = matrixA\r;

% Solves for b and d
for i = 1: n - 1
    coefficent(i, 3) =(coefficent(i + 1, 2) - coefficent(i, 2)) / (3 * deltaX(i));
    coefficent(i, 1) = dy(i) / deltaX(i) - deltaX(i) * (2 * coefficent(i, 2) +
coefficent(i + 1, 2)) / 3;
end

coefficent = coefficent(1: n - 1, 1: 3);
```