```
function [V, X, t] = myBlackScholes(r, sigma, X_0, X_T, K, T, M, N)
    % This function computes the solution to the Black-Scholes PDE using
    % the Crank-Nicolson finite difference method.
    %
    % INPUTS:
    % r:     risk-free interest rate
    % sigma: volatility of the underlying asset
    % X_0:   initial underlying asset price
    % X_T:   initial underlying asset price
    % K:     strike price
    % T:     time to maturity
    % M:     number of underlying asset price grid points
    % N:     number of time grid points
    %
    % OUTPUTS:
    % V:     option prices (M+1 x N+1 matrix)
    % X:     stock prices (M+1 x 1 vector)
    % t:     time points (N+1 x 1 vector)

    % Compute parameters
    dx = (X_T - X_0) / M;
    dt = T / N;
    u = @(p) 0.25 * dt * ((sigma^2 * p.^2) - (r * p));
    v = @(p) 0.5 * dt * ((sigma^2 * p.^2) + r);
    w = @(p) 0.25 * dt * ((sigma^2 * p.^2) + (r * p));

    % Initialize matrices
    V = zeros(M+1, N+1);
    X = (0:M)' * dx;
    t = (0:N)' * dt;

    % Set boundary conditions
    V(:, end) = max(X - K, 0);
    V(1, :) = 0;
    V(end, :) = max(X) - K * exp(-r * (T - t));

    % Set up tridiagonal matrix
    A = diag(1 + v(1:M-1)) - diag(u(2:M-1), -1) - diag(w(1:M-2), 1);
    B = diag(1 - v(1:M-1)) + diag(u(2:M-1), -1) + diag(w(1:M-2), 1);

    % Iterate through time steps backwards and compute call
    % option prices implicitly
    for n = N:-1:1
        % boundary set for previous time
        b = zeros(M-1,1);
        b(1) = -u(1) * V(1, n);
        b(end) = -w(M-1) * V(end, n);

        % boundary set for current time
```

```
        c = zeros(M-1,1);
        c(1) = u(1) * V(1, n);
        c(end) = w(M-1) * V(end, n);

        % solve for previous time
        V(2:M, n) = A \ ((B * V(2:M, n+1)) + c - b);

    end
end
```