```
function [LU,p,q,gf,L,U]= lucp(Ao)
%lupp  Computes the LU decomposition of A with partial pivoting
%
%   [LU,p,q,gf]= lucp(A) Computes the LU decomposition of A with complete
%   pivoting using vectorization. The factors L and U are returned in the
%   output A, and the permutation of the rows from partial pivoting are
%   recorded in the vector p. Here L is assumed to be unit lower
%   triangular, meaning it has ones on its diagonal. The resulting
%   decomposition can be extracted from A and p as follows:
%       L = eye(length(LU))+tril(LU,-1);     % L with ones on diagonal
%       U = triu(LU);                        % U
%       P = p*ones(1,n) == ones(n,1)*(1:n);  % Permutation matrix
%   A is then given as L*U = P*A, or P'*L*U = A.
%
%   output
%   ------
%   LU: matrix containing upper and unit lower triangular matrix
%   p: row permutation
%   q: column permutations
%   gf: growth factor of the matrix
%   Use this function in conjuction with backsub and forsub to solve a
%   linear system Ax = b.

A = Ao;
n = size(A,1);
p = (1:n);
q = (1:n);

for k=1:n-1
    % Find the row in column k that contains the largest entry in magnitude
    pos = max(max(abs(A(k:n,k:n))));
    [i,j] = find(abs(A(k:n,k:n)) == pos);

    row2swap = k-1+i(1);
    column2swap = k-1+j(1);

    %row swap
    A([k,row2swap],:) = A([row2swap,k],:);
    p([k,row2swap]) = p([row2swap,k]);
    %column swap
    A(:,[k,column2swap]) = A(:,[column2swap,k]);
    q([k,column2swap]) = q([column2swap,k]);
    % Perform the kth step of Gaussian elimination
    J = k+1:n;
    A(J,k) = A(J,k)/A(k,k);
    A(J,J) = A(J,J) - A(J,k)*A(k,J);
end

%unit lower triangular matrix L
L = eye(n);
L = L+tril(A,-1);

%upper triangular matrix U
U = triu(A,0);

LU = L*U;

%growth factor
```

```
u = abs(U);
%A = L*U;
a = abs(Ao);
u1 = max(u,[],'all');
a1 = max(a,[],'all');
gf = u1/a1;

end
```

```
Not enough input arguments.

Error in lucp (line 25)
A = Ao;
```