Problem 1

Part 1.
The following function takes in a matrix A in CSR data format and a dense vector x and returns the matrix multiplication $y = Ax$

```
% Assuming there are nzz nonzero entries
% val(1:nnz) contains the nonzero entries packed row by row,
  from left to right in each row
% colindex(1:nnz) contains the column index of each nonzero entry,
  i.e. val(i) is in column colindex(i) of A
% rowbegin(1:n+1) points to where each row begins: the nonzeros of row j
  are contained in val( rowbegin(j) : rowbegin(j+1)-1 )
function y = CSRMultiply(val, colIndex, rowBegin, x)
    n = length(rowBegin) - 1;
    y = zeros(n, 1);
    for row = 1 : n
        values = val(rowBegin(row): rowBegin(row + 1) - 1);
        colNums = colIndex(rowBegin(row): rowBegin(row + 1) - 1);
        for col = 1 : length(values)
            y(row) = y(row) + x(colNums(col)) * values(col);
        end
    end
end
```

We test `CSRMultiply` on random sparse matrix with random size using the following code

```
numSamples = 100;
errors = zeros(1, numSamples);

for i = 1 : numSamples
    % Generating random test matrices
    row = randi([1, 1000]);
    col = randi([1, 1000]);
    density = rand() * 1;
    A = sprand(row, col, density);
    [values, colIndex, rowBegin] = convertToCSR(full(A));
    x = rand(col, 1);

    % Comparing Matlab's built-in routine for y = A * x
    expected = A * x;
```

```
    computed = CSRMultiply(values, colIndex, rowBegin, x);
    errors(i) = sqrt(norm(expected - computed));
end
```

With matrix density value running from $0.2, 0.4, 0.6, 0.8, 1.0$. In each test, the absolute and relative error are all lower than machine epsilon (all 0s).

Part 2.
The following function instead takes in a matrix A in CSC data format and performs matrix multiplication.

```
function y = CSCMultiply(val, rowIndex, colBegin, x)
    n = length(colBegin) - 1;
    y = zeros(n, 1);
    for col = 1:n
        values = val(colBegin(col):colBegin(col + 1) - 1);
        rowIndices = rowIndex(colBegin(col):colBegin(col + 1) - 1);
        for i = 1:length(values)
            y(rowIndices(i)) = y(rowIndices(i)) + x(col) * values(i);
        end
    end
    y = y(y ~= 0);
end
```

The function is tested using the same script and similar results were obtained.

Problem 2 (Question 2.19)

1.
By Gershgorin's theorem, all eigenvalues λ of A lies within one of the discs $D(a_{ii}, \sum_{i\neq j}|a_{ij}|)$, in other words

$$|\lambda - a_{ii}| \leq \sum_{i\neq j}|a_{ij}| \implies a_{ii} - \sum_{i\neq j}|a_{ij}| \leq \lambda \leq a_{ii} + \sum_{i\neq j}|a_{ij}|$$

For some i. If $a_{ii} > 0$,

$$0 < |a_{ii}| - \sum_{i\neq j}|a_{ij}| = a_{ii} - \sum_{i\neq j}|a_{ij}| < \lambda$$

On the other hand, if $a_{ii} < 0$, then

$$\lambda < a_{ii} + \sum_{i\neq j}|a_{ij}| = -|a_{ii}| + \sum_{i\neq j}|a_{ij}| < 0$$

Since $a_{ii} \neq 0$, it follows that all eigenvalues λ is either stricly postive or negative. This suggests that

$$\det A = \prod_{i=1}^{k}\lambda_i \neq 0 \implies A \text{ is nonsingular}$$

2.
Since $|a_{11}| > |a_{j1}|$, the claim holds for base case.
After the first iteration,

$$\hat{a}_{ij} \leftarrow a_{ij} - \frac{a_{i1}a_{1j}}{a_{11}}$$

For $i > 1$. Since

$$\begin{aligned}
\sum_{i\neq j, j=2}^{n}|\hat{a}_{ij}| &\leq \sum_{i\neq j, j=2}^{n}|a_{ij}| + |\frac{a_{i1}a_{1j}}{a_{11}}| \\
&\leq |a_{ii}| - |a_{1i}| - |a_{1j}| + \frac{|a_{1j}|}{|a_{11}|}\sum_{i\neq j, j=1}^{n}|a_{i1}| \\
&\leq |a_{ii}| - |a_{1i}| \\
&\leq |a_{ii}| - |a_{1i}\frac{a_{i1}}{a_{11}}| \leq |\hat{a}_{ii}|
\end{aligned}$$

It follows that the Schur's complement is still strictly diagonally dominant. By performing the same argument to the Schur's complement in an inductive way, we see that partial pivoting does not permute any rows.