# Question 1

The functions are defined in the `math425hw2.m` file, under the section `%% Question 1`.

## Part A

See `math425hw2.m`.

## Part B

Using `myPartialPivot`, `myRank` simply counts the number of non-zero pivots in the upper triangular matrix returned from `myPartialPivot`.

Note that, even with partial pivoting, the computation still suffer from the precision issue of floating-point arithmetic. For this reason, in the implementation for `myRank`, I have rounded pivot entries to the tenth decimal place to determine if they are "true zeroes."

For example, consider a matrix

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}.$$

For the first pivot, we would row swap $R_3$ and $R_1$ and perform the row eliminations as follows.

$$\begin{pmatrix} 7 & 8 & 9 \\ 4 & 5 & 6 \\ 1 & 2 & 3 \end{pmatrix} \xrightarrow{-\frac{4}{7}R_1+R_2\to R_2} \begin{pmatrix} 7 & 8 & 9 \\ 0 & 3/7 & 6/7 \\ 1 & 2 & 3 \end{pmatrix} \xrightarrow{-\frac{1}{7}R_1+R_3\to R_3} \begin{pmatrix} 7 & 8 & 9 \\ 0 & 3/7 & 6/7 \\ 0 & 6/7 & 12/7 \end{pmatrix}$$

For the second pivot, no swap is necessary as $\frac{3}{7} > \frac{6}{7}$. So, we perform the row operation $-\frac{1}{2}R_2 + R_3 \to R_3$.

$$\begin{pmatrix} 7 & 8 & 9 \\ 0 & 6/7 & 12/7 \\ 0 & 3/7 & 6/7 \end{pmatrix} \xrightarrow{-\frac{1}{2}R_2+R_3\to R_3} \begin{pmatrix} 7 & 8 & 9 \\ 0 & 6/7 & 12/7 \\ 0 & 0 & 0 \end{pmatrix}$$

The above should be our upper triangular matrix, $U$. However, due floating point error, the result is not quiet zero. In MATLAB, when we display $U$ using `format rational`, it displays:

```
7          8          9
0          6/7        12/7
0          *          *
```

where the asterisks denote that the denominator is too large. By using `format long` we can see that the result is pretty much zero:

```
7.000000000000000   8.000000000000000   9.000000000000000
                0   0.857142857142857   1.714285714285714
                0   0.000000000000000   0.000000000000000
```

Using `format longG` we can see that the stored values aren't quiet zero. Hence, I have rounded the digits to ten decimal places.

```
   7                    8                    9
   0    0.857142857142857     1.71428571428571
   0  5.55111512312578e-17  1.11022302462516e-16
```

## Part C

Yes, the function virtually always returns 3. Since the question did not specify the bounds for the values from which a matrix could be generated, I simply used `rand(5, 3)` and `rand(3, 5)` to generate a $5 \times 3$ and $3 \times 5$ matrix, respectively. As such, it is virtually impossible for two or more rows to be identical or be a multiple of each other.

Furthermore, the floating-point precision issue discussed in part (b) would almost certainly ensure that the result appears to be full rank — especially when the values in the matrix are floating-point numbers to begin with — as values that should be zero may instead be interpreted as non-zero.