Programming Assignment - 3

Name:

In [1]:
```
#Import required packages here
import numpy as np
```

Question 1

Use the Gaussian elimination with scaled row-partial pivoting code to answer the following.

In [2]:
```
## Gaussian Elimination: Scaled Row Pivoting
## This function is based on the pseudo-code on page-148 in the Text by Kincaid and Cheney
def GE_rsp(A):
    '''
    This function returns the P'LU factorization of a square matrix A
    by scaled row partial pivoting.
    In place of returning L and U, elements of modified A are used to hold values of L and U.
    '''
    m,n = A.shape

    L = np.eye(n) # Not being used
    U = np.zeros_like(A) # Not being used
    if m !=n:
        sys.exit("This function needs a square matrix as an input.")

    # The initial ordering of rows
    p = list(range(n))

    # Scaling vector: absolute maximum elements of each row
    s = np.max(np.abs(A), axis=1)

    print("Scaling Vector: ",s)

    # Start the k-1 passes of Guassian Elimination on A
    for k in range(n-1):

        print("\n PASS {}: \n".format(k+1), A)
        # Find the pivot element and interchange the rows
        pivot_index = k + np.argmax(np.abs(A[p[k:], k])/s[p[k:]])

        # Interchange element in the permutation vector
        if pivot_index !=k:
            temp = p[k]
            p[k]=p[pivot_index]
            p[pivot_index] = temp
            print("permutation vector: ",p)

        print("\n Pivot Element: {0:.2f} \n".format(A[p[k],k]))
        if np.abs(A[p[k],k]) < 10**(-20):
             sys.exit("ERROR!! Provided matrix is non-singular.")

        # For the k-th pivot row Perform the Gaussian elimination on the following rows
        for i in range(k+1, n):
            # Find the multiplier
            z = A[p[i],k]/A[p[k],k]

            #Save z in A itself. You can save this in L also
            A[p[i],k] = z

            #Elimination operation: Changes all elements in a row simultaneously
```

```
            #Elimination operation: changes all elements in a row simultaneously
            ##
            A[p[i],k+1:] -= z*A[p[k],k+1:]
    return A, p
```

In [3]:

```
## Example on page number 146 (Kincaid Cheney).
## Example solved manually in class
A = np.array([[2, 3, -6], [1,-6,8], [3, -2, 1]], dtype=float)
print("\n Given A: \n ",A)
A,p =GE_rsp(A)
print("\n After Gaussian Elimination with RSPP: \n", A)
print("\n The permutation Vector is: \n", p)
```

```
 Given A: 
  [[ 2.  3. -6.]
 [ 1. -6.  8.]
 [ 3. -2.  1.]]
Scaling Vector:  [6. 8. 3.]

 PASS 1:
 [[ 2.  3. -6.]
 [ 1. -6.  8.]
 [ 3. -2.  1.]]
permutation vector:  [2, 1, 0]

 Pivot Element: 3.00


 PASS 2:
 [[ 0.66666667  4.33333333 -6.66666667]
 [ 0.33333333 -5.33333333  7.66666667]
 [ 3.         -2.          1.        ]]
permutation vector:  [2, 0, 1]

 Pivot Element: 4.33


 After Gaussian Elimination with RSPP: 
 [[ 0.66666667  4.33333333 -6.66666667]
 [ 0.33333333 -1.23076923 -0.53846154]
 [ 3.         -2.          1.        ]]

 The permutation Vector is: 
 [2, 0, 1]
```

> - (A) Modify this code to write a function that solves a linear system Ax =b. Test this in the case when $b = [3, 1, 1]^T$, and the matrix $A = [1\ 6\ 0; 2\ 1\ 0; 0\ 2\ 1]$. Only display the solution in the output.

In [ ]:

```
# Your code come here
```

> - (B) Modify this code to find the determinant of any square matrix A. Note that
>
> $$PA = LU \Rightarrow \det A = \pm \det U.$$
>
> The sign depends of the number of row-swaps in the elimination process. Use this code to find the determinant of any 10×10 matrix that you randomly generate. Compare your result with the built-in NumPy method.

In [ ]:

```
#Your code comes here
```

> - (C) Modify the system-solver that you have created to find the inverse of a square matrix. Use this code to display the inverse of A = [1 6 0; 2 1 0; 0 2 1].

In [ ]:

```
# Your code comes here
```