In [64]:
```
import numpy as np

def get_rref(arr):

    def swap_rows(arr, row1, row2):
        arr[[row1, row2]] = arr[[row2, row1]]

    def scale_row(arr, row, scalar):
        new_row = arr[row] *scalar
        arr[row] = new_row.astype(float)

    def add_row(arr, source_row, target_row, scalar=1):
        arr[target_row] += scalar * arr[source_row]

    def find_pivot_row(arr, col):
        num_rows = arr.shape[0]
        for row in range(col, num_rows):
            if arr[row, col] != 0:
                return row
        return None

    def eliminate_below(arr, pivot_row, pivot_col):
        num_rows, num_cols = arr.shape
        pivot_val = arr[pivot_row, pivot_col]
        for row in range(pivot_row + 1, num_rows):
            factor = arr[row, pivot_col] / pivot_val
            arr[row, pivot_col] = 0
            for col in range(pivot_col + 1, num_cols):
                arr[row, col] -= factor * arr[pivot_row, col]

    def eliminate_above(arr, pivot_row, pivot_col):
        for row in range(pivot_row - 1, -1, -1):
            factor = arr[row, pivot_col]
            arr[row, pivot_col] = 0
            for col in range(pivot_col + 1, arr.shape[1]):
                arr[row, col] -= factor * arr[pivot_row, col]

    def reduce_to_rref(arr):
        num_rows, num_cols = arr.shape
        pivot_col = 0
        for row in range(num_rows):
            if pivot_col >= num_cols:
                break
            pivot_row = find_pivot_row(arr, pivot_col)
            if pivot_row is None:
                pivot_col += 1
                continue
            swap_rows(arr, row, pivot_row)
            scale_row(arr, row, 1 / arr[row, pivot_col])
            eliminate_below(arr, row, pivot_col)
            eliminate_above(arr, row, pivot_col)
            pivot_col += 1
            check_all_zero(arr)

    def check_all_zero(arr):
        zero_rows = np.where(~arr.any(axis=1))[0]
        if zero_rows.size > 0:
            last_row = arr.shape[0] - 1
```

```
            for row in zero_rows:
                if row != last_row:
                    swap_rows(arr, row, last_row)
                    last_row -= 1

    reduce_to_rref(arr)
    arr = arr.astype(float)
    return arr
```

In [65]:
```
np.random.seed(42)
arr1 = np.random.randint(0, 20, (5, 5)).astype(float)
arr2 = np.random.randint(0, 20, (3, 3)).astype(float)
arr3 = np.random.randint(0, 20, (6, 4)).astype(float)
arr4 = np.random.randint(0, 20, (3, 5)).astype(float)
arr5 = np.random.randint(0, 20, (2, 2)).astype(float)
arr6 = np.random.randint(0, 100, (5, 5)).astype(float)
arr7 = np.random.uniform(0, 40, size=(5, 5)).astype(float)
arr8 = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]], dtype=np.float64)
test_cases = [arr1, arr2, arr3, arr4, arr5, arr6, arr7, arr8]
```

In [66]:
```
for index,matrix in enumerate(test_cases):
    print(f"test matrix : {index+1}")
    print("Original Matrix :")
    print(matrix)
    print("Reduced row echolon form : ")
    print(get_rref(matrix))
    print("-----------------------------------------------------------------------------
```

```
test matrix : 1
Original Matrix :
[[ 6. 19. 14. 10.  7.]
 [ 6. 18. 10. 10.  3.]
 [ 7.  2.  1. 11.  5.]
 [ 1.  0. 11. 11. 16.]
 [ 9. 15. 14. 14. 18.]]
Reduced row echolon form :
[[ 1.  0.  0.  0.  0.]
 [-0.  1.  0.  0.  0.]
 [ 0.  0.  1.  0.  0.]
 [ 0.  0.  0.  1.  0.]
 [ 0.  0.  0.  0.  1.]]
------------------------------------------------------------------------------------
------------
test matrix : 2
Original Matrix :
[[11. 19.  2.]
 [ 4. 18.  6.]
 [ 8.  6. 17.]]
Reduced row echolon form :
[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
------------------------------------------------------------------------------------
------------
test matrix : 3
Original Matrix :
[[ 3. 13. 17.  8.]
 [ 1. 19. 14.  6.]
 [11.  7. 14.  2.]
 [13. 16.  3. 17.]
 [ 7.  3.  1.  5.]
 [ 9.  3. 17. 11.]]
Reduced row echolon form :
[[ 1.  0.  0.  0.]
 [ 0.  1.  0.  0.]
 [-0. -0.  1.  0.]
 [ 0.  0.  0.  1.]
 [ 0.  0.  0.  0.]
 [ 0.  0.  0.  0.]]
------------------------------------------------------------------------------------
------------
test matrix : 4
Original Matrix :
[[ 1.  9.  3. 13. 15.]
 [14.  7. 13.  7. 15.]
 [12. 17. 14. 12.  8.]]
Reduced row echolon form :
[[   1.           0.           0.          46.28571429  104.85714286]
 [  -0.           1.           0.          15.52380952   33.23809524]
 [   0.           0.           1.         -57.66666667 -129.66666667]]
------------------------------------------------------------------------------------
------------
test matrix : 5
Original Matrix :
[[14. 12.]
 [ 0.  6.]]
Reduced row echolon form :
[[1. 0.]
```

```
 [0. 1.]]
--------------------------------------------------------------------------------
------------
test matrix : 6
Original Matrix :
[[ 8. 87.  0.  7. 87.]
 [62. 10. 80.  7. 34.]
 [34. 32.  4. 40. 27.]
 [ 6. 72. 71. 11. 33.]
 [32. 47. 22. 61. 87.]]
Reduced row echolon form :
[[ 1.  0.  0.  0.  0.]
 [-0.  1.  0.  0.  0.]
 [-0. -0.  1.  0.  0.]
 [ 0.  0.  0.  1.  0.]
 [ 0.  0.  0.  0.  1.]]
--------------------------------------------------------------------------------
------------
test matrix : 7
Original Matrix :
[[26.63689426 23.65191151 10.98887172 22.44973703 15.31707499]
 [38.86848382 33.95655297 28.86918085  9.43939679 10.24273291]
 [ 1.61734358 28.42651559  4.43563283 17.57346007  8.06876809]
 [35.83054383 19.01480893 22.53102288 27.82064346  5.57325818]
 [24.17669517 21.59364365  8.12244899 37.71414282 23.95461866]]
Reduced row echolon form :
[[ 1.  0.  0.  0.  0.]
 [-0.  1.  0.  0.  0.]
 [ 0.  0.  1.  0.  0.]
 [ 0.  0.  0.  1.  0.]
 [ 0.  0.  0.  0.  1.]]
--------------------------------------------------------------------------------
------------
test matrix : 8
Original Matrix :
[[1. 2. 3.]
 [4. 5. 6.]
 [7. 8. 9.]]
Reduced row echolon form :
[[ 1.  0. -1.]
 [-0.  1.  2.]
 [ 0.  0.  0.]]
--------------------------------------------------------------------------------
------------
```

In [ ]: