In [1]:

```
import numpy as np
```

Problem 1

Solving for row echelon form (GF2)

In [2]:

```
rowlist = np.array([[1,1,0,1],
                    [1,0,1,0],
                    [0,1,1,1],
                    [0,1,1,0]])
```

In [85]:

```
def ech_rowGF2(rowlist):
    col_label_list = np.array([a for a in range(len(rowlist[0]))])
    rows_left = set(range(len(rowlist)))
    new_rowlist = []
    for c in col_label_list:
        #among rows left, list of row-labels whose rows have a nonzero in positio
        rows_with_nonzero = [r for r in rows_left if rowlist[r][c] != 0]
        if rows_with_nonzero != []:
            pivot = rows_with_nonzero[0]
            rows_left.remove(pivot)
            new_rowlist.append(rowlist[pivot] % 2)
            for r in rows_with_nonzero[1:]:
                multiplier = rowlist[r][c] / rowlist[pivot][c]
                rowlist[r] = rowlist[r] - multiplier * rowlist[pivot]
    new_rowlist = np.array(new_rowlist)
    return new_rowlist
```

In [4]:

```
print(ech_rowGF2(rowlist))
```

```
[[1 1 0 1]
 [0 1 1 1]
 [0 0 0 0]
 [0 0 0 1]]
```

Problem 2

Solving system of equations through Gauss-Jordan [RREF] elimination method

In [105]:

```
def ech_row(rowlist):
    col_label_list = np.array([a for a in range(len(rowlist[0]))])
    rows_left = set(range(len(rowlist)))
    new_rowlist = []
    for c in col_label_list:
        #among rows left, list of row-labels whose rows have a nonzero in positio
        rows_with_nonzero = [r for r in rows_left if rowlist[r][c] != 0]
        if rows_with_nonzero != []:
            pivot = rows_with_nonzero[0]
            rows_left.remove(pivot)
            new_rowlist.append(rowlist[pivot])
            for r in rows_with_nonzero[1:]:
                multiplier = rowlist[r][c] / rowlist[pivot][c]
                rowlist[r] = rowlist[r] - multiplier * rowlist[pivot]
    new_rowlist = np.array(new_rowlist)
    return new_rowlist
```

In [405]:

```
def rref(rowlist):
    col_label_list = np.array([a for a in range(len(rowlist[0]))])
    rows_left = set(range(len(rowlist)))
    new_rowlist = []
    for i in col_label_list:
        rows_with_nonzero = [r for r in rows_left if rowlist[r][i] != 0]
        if rows_with_nonzero != []:
            pivot = rows_with_nonzero[0]
            rows_left.remove(pivot)
            for r in rows_with_nonzero:
                    if rowlist[r][pivot] > 1 or rowlist[r][pivot] < 0:
                        multiplier = rowlist[r][pivot] / 1
                        rowlist[r] = rowlist[r] / multiplier
                    elif rowlist[r][pivot] == 0:
                        if rowlist[r][pivot + 1] > 1 or rowlist[r][pivot + 1] < 0
                            multiplier = rowlist[r][pivot + 1] / 1
                            rowlist[r] = rowlist[r] / multiplier
                    new_rowlist.append(rowlist[r])

    new_rowlist = np.array(new_rowlist)

    col_list = [a for a in range(len(rowlist[0]) - 1)]
    rows = [a for a in range(len(rowlist))]
    rref = []
    for i in col_list:
        nonzero_rows = [c for c in rows if new_rowlist[c][i] != 0]
        if nonzero_rows != []:
            row = nonzero_rows[len(nonzero_rows) - 1]
            pivot = new_rowlist[row][i]
            if len(nonzero_rows) > 1:
                for b in nonzero_rows:
                    if row != b:
                        multiplier = pivot * new_rowlist[b][i]
                        new_rowlist[b] = new_rowlist[b] - (multiplier * new_rowli
    return new_rowlist
```

a)

In [36]:

```
array = np.array([[4,2,3,4,5],
                  [2,0,0,3,2],
                  [3,2,3,4,5],
                  [2,0,0,6,7]])
b = np.array([[2], [3], [4], [5]])
rowlist = np.hstack((array, b))
rowlist = rowlist.astype(np.float)
```

In [406]:

```
ech = ech_row(rowlist)
print(ech)
print(rref(ech))
```

```
[[ 4.   2.   3.   4.   5.   2. ]
 [ 0.  -1.  -1.5  1.  -0.5  2. ]
 [ 0.   0.   0.   1.5  1.   3.5]
 [ 0.   0.   0.   0.   3.  -5. ]]
[[ 1.          0.          0.          0.          0.         -2.        ]
 [ 0.          1.          1.5         0.          0.          2.27777778]
 [ 0.          0.          0.          1.          0.          3.44444444]
 [ 0.          0.          0.          0.          1.         -1.66666667]]
```

After transforming the row echelon form of the augmented matrix, the following is the solution for the system:

X1 = -2

X2 + 1.5X3 = 2.27..

X4 = 3.44..

X5 = -1.66..

b)

In [409]:

```
array = np.array([[4,2,3,4,5],
                  [2,0,0,3,2],
                  [3,2,3,4,5],
                  [2,0,0,6,7]])
b = np.array([[2], [3], [4], [5]])
rowlist = np.hstack((array, b))
rowlist = rowlist.astype(np.float)
print(ech_row(rowlist))
```

```
[[ 4.   2.   3.   4.   5.   2. ]
 [ 0.  -1.  -1.5  1.  -0.5  2. ]
 [ 0.   0.   0.   1.5  1.   3.5]
 [ 0.   0.   0.   0.   3.  -5. ]]
```

Above is the row reduced form of the system of equations, able to work with floats