Programming Assignment #2
Gauss-Jordan Elimination

The purpose of this program was to be able to take a system of equations which would be entered as a matrix of values.

For the Gauss-Jordan program we follow the steps as seen in the code below to get the desired result. The desired result is to check through the matrix and find the max value in the rows and then make swaps for the largest absolute value to be on the diagonal. Then we divide the entire row to get the value on the diagonal to a value of one. Then go through the matrix rows and subtract every value by the values in the row that we made the diagonal value one and then multiply the diagonal value before the subtraction occurs on each row. That gets repeated until the values on the diagonal are all equal to the value of 1. This is all shown in the code below:

```
# CSC 340
# Programmig assignmnet #3

# Kareem Tiwari

##
## create an augmented matrix
##Mat=[[1,-2,0,-4],
##     [3,1,-2,1],
##     [0,-2,2,2]]
Mat = [[1,0,2, 1],[2,-1,3,-1],[4,1,8,2]]
##Mat = [[1,-1,0,1,0,0],
##       [2,0,4,0,1,0],
##       [0,2,-1,0,0,1]] #Use this one  to show the inverse
##Mat = [[1,0,0,0],[1,1,1,1],[1,2,4,8],[1,3,9,27]]
##print(Mat)
#his  is a parameter to check if the algorithm successfully found a unique solution
#This loop finds the greatest magnitude in column one
#Column one
# this is the swap,multiplication, adn subtraction all for column one
E=1

for j in range(len(Mat)):
    maxVal = 0
    maxPos = 0
    for i in range(j,len(Mat)):
       if abs(Mat[i][j]) > maxVal: # find the max value
           maxVal = abs(Mat[i][j])
           maxPos = i
    temp = Mat[maxPos]  # swap values
    Mat[maxPos] = Mat[j]
    Mat[j] = temp
    dividend = Mat[j][j]  # Divide by the value on the diagonal
    for i in range(len(Mat[0])):

       Mat[j][i] = Mat[j][i]/(dividend) #divide every value by the value on the diagonal


    for a in range(len(Mat)):
        Mult = Mat[a][j]
        if a != j:

            for i in range(len(Mat[0])):


                Mat[a][i] = Mat[a][i]- (Mat[j][i]*Mult)

print(Mat)


                # if Mat[i][j] == 1:
                    # print("The value on the diagonal is one")
                # else:
                    # print("The value on the diagonal is NOT one, something isn't right")
```

As well as creating the Gauss-Jordan program find the identity matrix of system of equations given. We were also challenged to create a Gauss-Jordan program that id dynamic enough to find the inverse of the matrix that is input into the program. The code that I had shown previously is dynamic enough to have the matrix input to find the matrix inverse.

The code implementation as well as the result is below:

```
# CSC 340
# Programmig assignmnet #3

# Kareem Tiwari

##
## create an augmented matrix
##Mat=[[1,-2,0,-4],
##      [3,1,-2,1],
##      [0,-2,2,2]]
##Mat = [[1,0,2, 1],[2,-1,3,-1],[4,1,8,2]]
Mat = [[1,-1,0,1,0,0],
       [2,0,4,0,1,0],
       [0,2,-1,0,0,1]] #Use this one  to show the inverse
##Mat = [[1,0,0,0],[1,1,1,1],[1,2,4,8],[1,3,9,27]]
##print(Mat)
#his  is a parameter to check if the algorithm successfully found a unique solu
#This loop finds the greatest magnitude in column one
#Column one
# this is the swap,multiplication, adn subtraction all for column one
E=1

for j in range(len(Mat)):
    maxVal = 0
    maxPos = 0
    for i in range(j,len(Mat)):
       if abs(Mat[i][j]) > maxVal: # find the max value
           maxVal = abs(Mat[i][j])
           maxPos = i
    temp = Mat[maxPos]  # swap values
    Mat[maxPos] = Mat[j]
    Mat[j] = temp
    dividend = Mat[j][j]  # Divide by the value on the diagonal
    for i in range(len(Mat[0])):

       Mat[j][i] = Mat[j][i]/(dividend) #divide every value by the value on the


    for a in range(len(Mat)):
        Mult = Mat[a][j]
        if a != j:

            for i in range(len(Mat[0])):


                Mat[a][i] = Mat[a][i]- (Mat[j][i]*Mult)

print(Mat)
```

```
= RESTART: /Volumes/LaCie/CSC340-Spring2022/Programming Assignment#2/Gaus-Jordan
(2022).py
[[1.0, 0.0, 0.0, 0.8, 0.09999999999999998, 0.4], [0.0, 1.0, 0.0, -0.2, 0.1, 0.4]
, [-0.0, -0.0, 1.0, -0.4, 0.2, -0.2]]
>>> |
```

```
##
## create an augmented matrix
##Mat = [[9, -4], [2],
##       [-3], [0], [6],
##       [3], [1], [3]]
##Mat =[[9,-4,2],[-3,0,6],[3,1,3]]
##Mat = [[1,0,0,0,-3],[1,1,1,1,0],[1,2,4,8,5],[1,3,9,27,18]]
Mat = [[1,0,2, 1],[2,-1,3,-1],[4,1,8,2]]
E=1

print(Mat)

for j in range(len(Mat)): # itterate throught the columns in the matrix
    maxVal = 0
    maxPos = 0
    for i in range(j,len(Mat)): #
       if abs(Mat[i][j]) > maxVal: #find the max value
           maxVal = abs(Mat[i][j])
           maxPos = i
    temp = Mat[maxPos]
    Mat[maxPos] = Mat[j] # swap values
    Mat[j] = temp
    dividend = Mat[j][j] #set the dividend

    for a in range(j+1, len(Mat)):
        Mult = Mat[a][j]
        if a != j:

            for i in range(len(Mat[0])):


                Mat[a][i] = Mat[a][i]- (Mat[j][i]*Mult/dividend)

print(Mat)
print(Mat[a][i])
```

For the Gaussian elimination program, we follow just about the same steps:

First, we iterate through the columns in the matrix and then we also found the max value in the column itself. Then we would swap the max value to get it on the diagonal. Next, we would divide to get the ones on the diagonal and to make the upper triangular matrix with the lower triangle being where the zeros reside. We would continue this until we have an upper triangular matrix.

Finally, we created a program that finds the matrix determinant following the Gaussian elimination method. In this program first, we create a counter and then just as the Gaussian elimination we iterate through the columns to find the max value then we swap the rows if the max value is found not on the diagonal. We then zero out all of the values that fall under the diagonal creating the. Upper triangular matrix that is desired.

```python
#Guassian Elimination Matrix Determinant


##Mat = [[1,0,2, 1],[2,-1,3,-1],[4,1,8,2]]
Mat = [[1,-1,0],[-2,2,-1],[0,1,-2]]
r = 0 # counter
E = 1

for j in range(len(Mat)):
     maxVal = 0
     maxPos = 0
     for i in range(j,len(Mat)): #
          if abs(Mat[i][j]) > maxVal: #find the max value

               maxVal = abs(Mat[i][j])
               maxPos = i
     if maxPos != j:
          temp = Mat[maxPos]
          Mat[maxPos] = Mat[j] # swap values
          Mat[j] = temp
          dividend = Mat[j][j] #set the dividend
          r += 1

     for a in range(j+1, len(Mat)):
        Mult = Mat[a][j]
        if a != j:

             Mat_det =  (-1**r)* (Mat[j][i]*Mult/dividend)
print(Mat)
print(Mat_det)
```