5.1 Give the result each of the following matrix operations.

a. 4 \* =

b. + =

c.- =

d. *T* =

e. 1/3\*(*T*) = 1/3\* =

f. = Can’t be done!! Need same XxY

5.2 Which of the following matrix multiplications cannot be performed?

a. \* = YES

b. [ ]\* = YES

c. \* = NO In order to perform the function the number of columns in the first matrix must equal the number of rows in the second matrix

d. \**T* = YES

e.\* = YES

f. \* = NO In order to perform the function the number of columns in the first matrix must equal the number of rows in the second matrix

5.3 Perform all valid matrix multiplications from the previous question.

a. \* = (1\*-1)+(0\*0) : (1\*3)+(0\*-2) : (2\*-1)+(1\*0) : (2\*3)+(1\*-2) =

b. [ ]\* = (2\*-1)+(3\*3)+(1\*1) : (2\*0)+(3\*1)+(1\*1) : (1\*0)+(3\*-2)+(1\*0) = [ ]

c. NOT DONE

d. \**T* = \* = (1\*1)+(-2\*-2)+(2\*2) : (1\*4)+(-2\*-1)+2\*0) : (4\*1)+(-1\*-2)+(0\*2) : (4\*4)+(-1\*-1)+(0\*0) =

e. \* = (1\*1)+(0\*4)+(-1\*1) : (1\*2)+(0\*-1)+(-1\*3) : (2\*1)+(1\*4)+(0\*1) : (2\*2)+(1\*-1)+(0\*3) =

f. NOT DONE

5.4 Compute the determinate of the following matrices. Indicate which matrices

are singular.

a. = (1\*1)-(2\*0) = 1

b. = (2\*6)-(9\*3) = 9

c. =(4\*-1)-(-2\*-2) = 0 (SINGULAR)

d. =. =(1\*2\*3)+(0\*6\*2)+(2\*4\*1) –(2\*2\*2)-(1\*6\*1)-(3\*4\*0) = (6+12+8)+(-8-6-12) = 0 (SINGULAR)

5.5 Find the inverse for all non-singular matrices from the previous problem.

a. : = \*row 1 by -2 and add to row 2 =

b. : = /row 1 by 2 = : = \*row 1 by -9 add to row 2 = : = \*row 2 by -2/15 = : = \*row 2 by -3/2 add to row 1 = :

6.1 Convert the following polar coordinate points to the 2D Euclidean coordinate system. x =  cos  and y =  sin.

a.  =5,= = x= 5cos, y=5sin = x=0, y=5 = [0,5]

b. =5,= = x=-5, y=0 = [-5,0]

c. =5,= = x=0, y=-5 = [0,-5]

d. =5,=m = x=5, y=0 = [5,0]

e. =,= = x=1, y=1 = [1,1]

f. =,= = x=-2, y=2 [-2,2]

g. =10,= = x=7.66, y= 6.42 [7.66,6.42]

h. =3,= = x=1.14,y=2.77 [1.14,2.77]

6.3 Construct and show the 2D homogeneous affine transformation matrix to perform

the following operations.

a. Rotate by around the origin. =

b. Translate by -2 along the y-axis and 1 along the x-axis.

c. Scale by ½ along the y-axis. =

d. Perform all three of the above simultaneously. USED formula: =

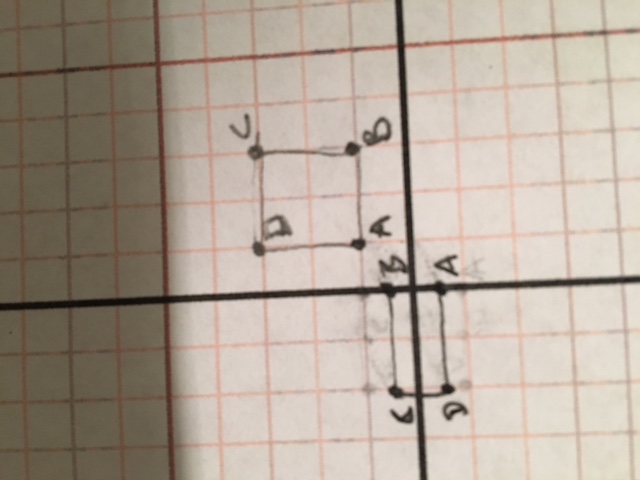
=

6.4 Sketch the square described by the vertices below. Re-express these four vectors

in homogeneous coordinates. Perform the transformation from exercise 5.3 d

on all four vertices. Sketch the resulting matrix.

= =



Python:

*'''  
  
Write and submit a python script that does the following:  
 -Reads in a file of numbers where each row contains x, y, and z coordinate values for a point and there are an arbitrary number of points  
 -Uses python linear algebra mechanisms to perform the following affine transformations:  
 -rotate by 0.64 radians  
 -translate along the x-axis by 2.3 and the y axis by -0.7  
 -scale along the y-axis by 0.75  
 -Prints each of the new points after these transformations  
  
'''***import** numpy **as** np  
*#Save text file in the same folder as the Python code. Text file is the matrix in list form.  
#change name of file for additional matrix*myFile = open(**'matrix\_file.txt'**)  
  
*#Python to read text file*myTextLines = myFile.readlines()  
myCoords = []  
**for** textline **in** myTextLines:  
 rowItems = textline.strip().split(**','**)  
 rowValues = []  
 **for** item **in** rowItems:  
 rowValues.append( float(item) )  
 myCoords.append(rowValues)  
  
*#define your variables*sX = float(input(**"Enter Value to Scale X:"**))  
  
sY = float(input(**"Enter Value to Scale Y:"**))  
  
alpha = float(input(**"Enter Value to Rotate (in radinas):"**))  
*#rotation vlaue must be at a real intager (can be listed as pi halves)*tX = float(input(**"Enter Value to Translate X:"**))  
  
tY = float(input(**"Enter Value to Translate Y:"**))  
  
  
*#sX = 1  
#sY = .75  
#alpha = 0.64  
#tX = 2.3  
#tY = -0.7***'''  
[[sX\*cos\*alpha, sX\*-sin\*alpha, sX\*tX]  
 [sY\*sin\*alpha, sY\*cos\*alpha, sY\*tY]  
 [0,0,1]]  
  
'''***#do operations here***for** coord **in** myCoords:  
 A = np.matrix([[sX\*np.cos(alpha),sX\*-(np.sin(alpha)),sX\*tX],[sY\*np.sin(alpha),sY\*np.cos(alpha),sY\*tY],[0,0,1]])  
 M = np.matrix([coord])  
 print(**"New Transformation for: "**, M)  
 print(A\*M.T)  
 print()