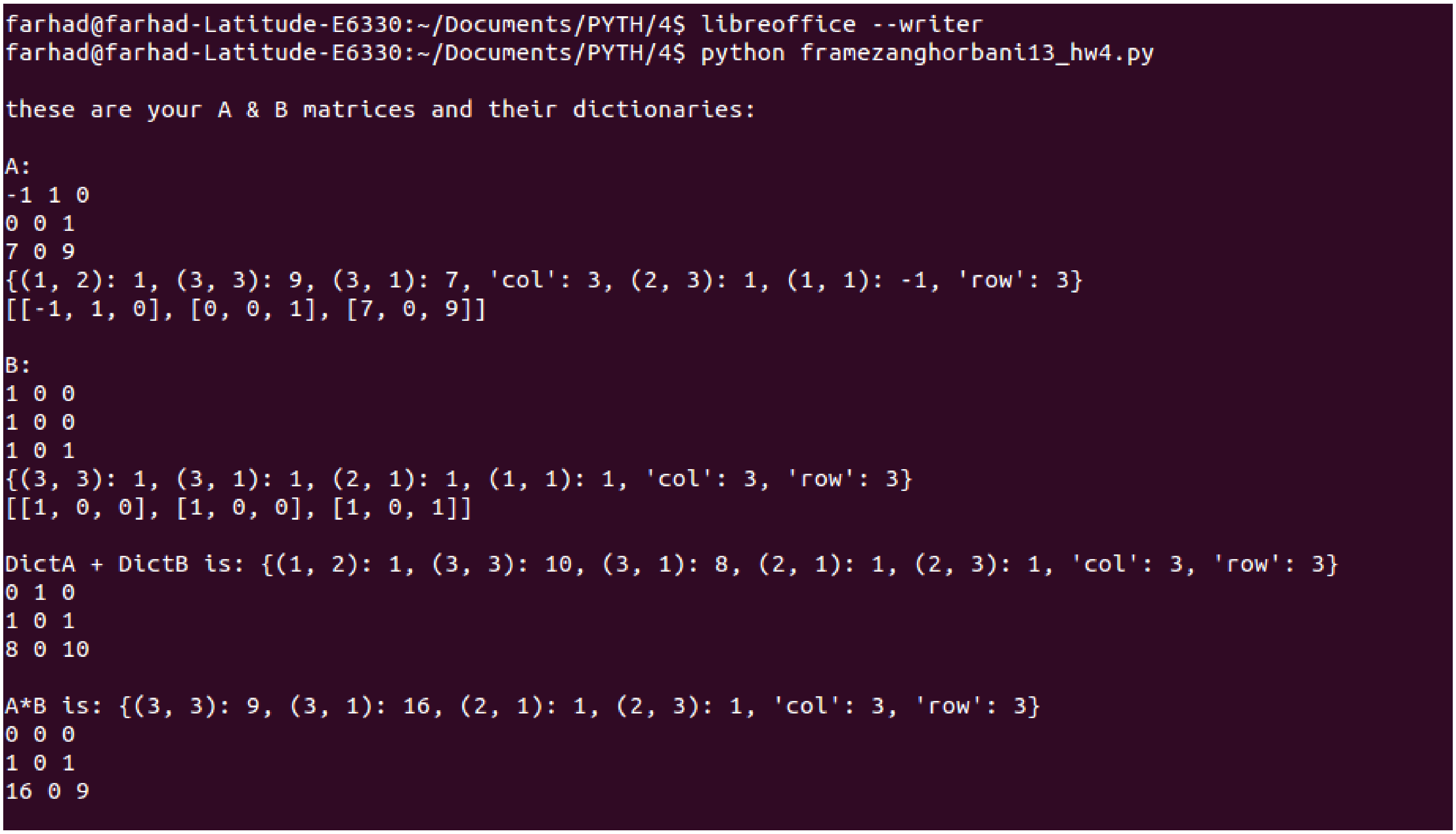
Farhad Ramezanghorbani HW4

```
#function###########################################
def matrix_import():                                    # this function will get column and
    Listoflist = []                                     # row of matrix from user then it will
    print "enter matrix dimensions"                     # get each of matrix elements from
    rowm = int(raw_input("rows: "))                     # user and put them in a list of list
    colm = int(raw_input("columns: "))

    print "\nenter matrix elements"
    for i in range(rowm):
        List = []
        for j in range(colm):
            xm = int(raw_input("enter %d,%d element of matrix: " %(i+1,j+1)))
            List.append(xm)
        Listoflist.append(List)
    return Listoflist

#function###########################################
def matrix_represent(M):                                # this function will show matrix
    for row in M:                                       # representation of a list
        for element in row:
            print element,
        print

#function###########################################
def convert_todict(L):                                  # this function will convert a list
    dictL = {}                                          # to a dictionary in which there is
    rowL = len(L)                                       # no key with zero value. it has the
    colL = len(L[0])                                    # row and column of sparse matrix
    for i in range(rowL):
        for j in range(colL):
            if L[i][j] != 0:
                dictL[i+1,j+1] = L[i][j]
    dictL['row'] = rowL
    dictL['col'] = colL
    return dictL

#function###########################################
def convert_tolist(D):                                  # this function will convert a dict.
                                                        # of non zero elements of sparse matrix
    Listoflist = []                                     # to a list in which we have the zero
    rowL = D['row']                                     # elements also, then we can see the
    colL = D['col']                                     # matrix representation of this list
    for i in range(rowL):
        List = []
        for j in range(colL):
            if D.has_key((i+1,j+1)) == 1:
                List.append(D[i+1,j+1])
            else:
                List.append(0)
        Listoflist.append(List)
    return Listoflist

#function###########################################
def addsparse_dict(Da,Db):                              # in matrix summation for less calculation
    Dc = {}                                             # it is better to omit the zero elements of
    rowa = Da['row']                                    # matrices, it means that we can use sparse
    cola = Da['col']                                    # dictionaries for calculation. this function
    rowb = Db['row']                                    # will add two sparse matrix
    colb = Db['col']
    if (rowa == rowb and cola == colb):
        Dc['row'] = rowa
        Dc['col'] = cola
        for i in range(rowa):
            for j in range(cola):

                if (Da.has_key((i+1,j+1)) == 1      and Db.has_key((i+1,j+1)) == 1):
                    Dc[i+1,j+1] = Da[i+1,j+1] + Db[i+1,j+1]
                elif (Da.has_key((i+1,j+1)) == 1):
                    Dc[i+1,j+1] = Da[i+1,j+1]
                elif (Db.has_key((i+1,j+1)) == 1):
                    Dc[i+1,j+1] = Db[i+1,j+1]
                if (Dc.has_key((i+1,j+1)) == 1):
                    if (Dc[i+1,j+1] == 0):
                        del Dc[i+1,j+1]

        return Dc
    else:
        print "\nfor addition two matrices should have same dimentions!"

#function########################################
def multiplyab_dict(Da,Db):                             # this function uses dictionary for matrix
    Dc = {}                                             # multiplication
    if (Da['col'] == Db['row']):
        Dc['row'] = Da['row']
        Dc['col'] = Db['col']
        for i in range(Da['row']):
            for j in range(Db['col']):
                c = 0
                for k in range(Da['col']):
                    if (Da.has_key((i+1,k+1)) == 1 and Db.has_key((k+1,j+1)) == 1):
                        c += Da[i+1,k+1]*Db[k+1,j+1]
                if (c!=0):
                    Dc[i+1,j+1] = c

        return Dc
    else:
        print "\nfor multiplication #of column of A  should be equal to #of row of B!"

################################################

#print "import matrix A:"                               # generally we should use these four lines
#A = matrix_import()                                    # but for just seeing the result I wrote two
#print "\nimport matrix B:"                             # list as A and B matrices with same dimention
#B = matrix_import()                                    # of 3x3 in following lines

##for test
A = [[-1,1,0],[0,0,1],[7,0,9]]
B = [[1,0,0],[1,0,0],[1,0,1]]

print "\nthese are your A & B matrices and their dictionaries:\n\nA:"
matrix_represent(A)                                     # matrix representation of A
dictA = convert_todict(A)                               # convert nonzero values of A to a dictionary type
print dictA
Lista = convert_tolist(dictA)
print Lista                                             # show the list representation

print "\nB:"
matrix_represent(B)
dictB = convert_todict(B)
print dictB
Listb = convert_tolist(dictB)
print Listb

print
dictC = addsparse_dict(dictA,dictB)                     # add A and B (dictionaries)
print "DictA + DictB is:",dictC                         # show the dict. representation
if (dictC != None):                                     # for the case that A and B have not same dimension
```

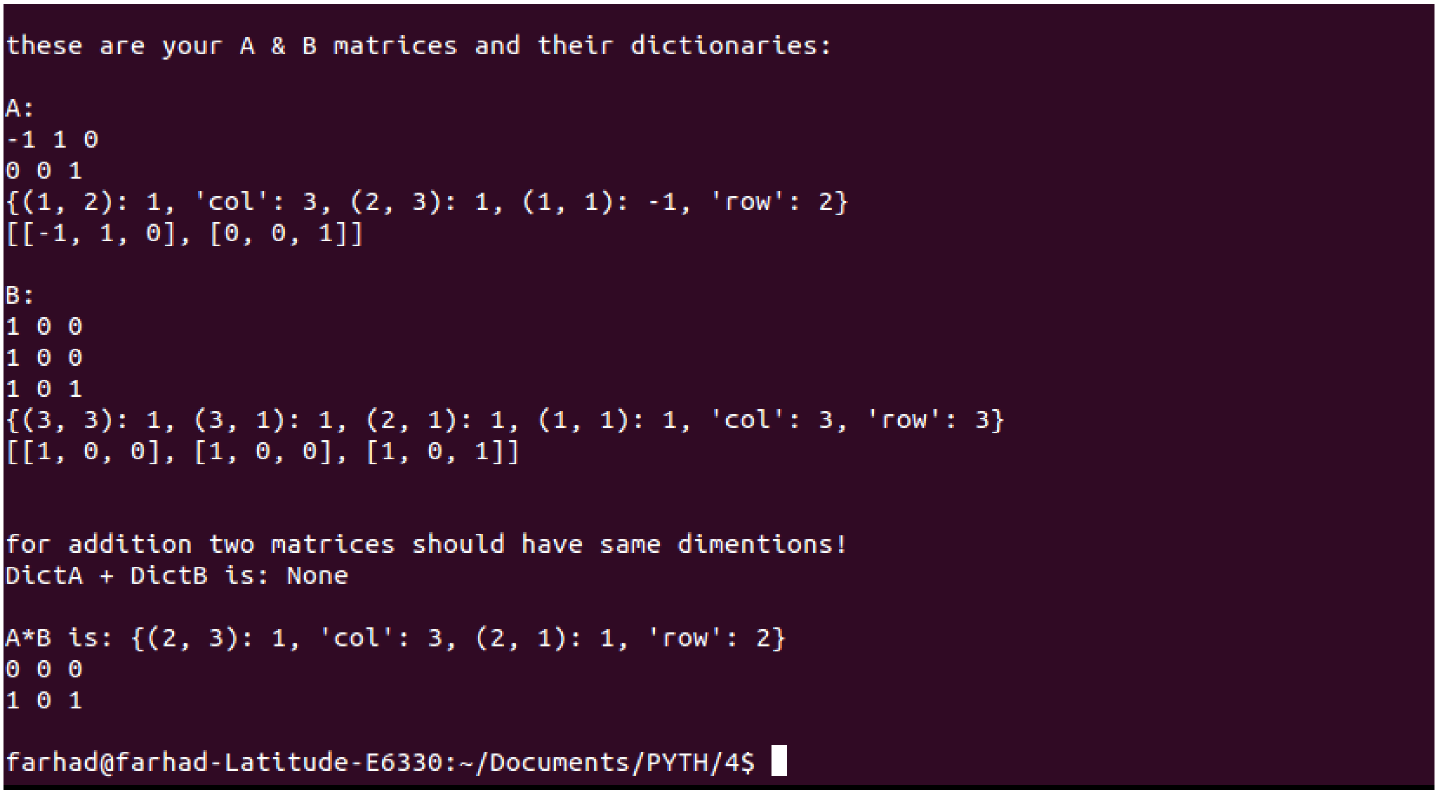
```
        ListC = convert_tolist(dictC)                  # convert the result to list to see the matrix repres.
        matrix_represent(ListC)
print

dictD = multiplyab_dict(dictA,dictB)                   # multiply A and B (dictionaries)
print "A*B is:", dictD                                 # show the dict. representation
if (dictD!=None):                                      # for the case that columnA != rowB
        ListD = convert_tolist(dictD)
        matrix_represent(ListD)

print

################################################
```

This is the result for sample input for 3x3 matrices:



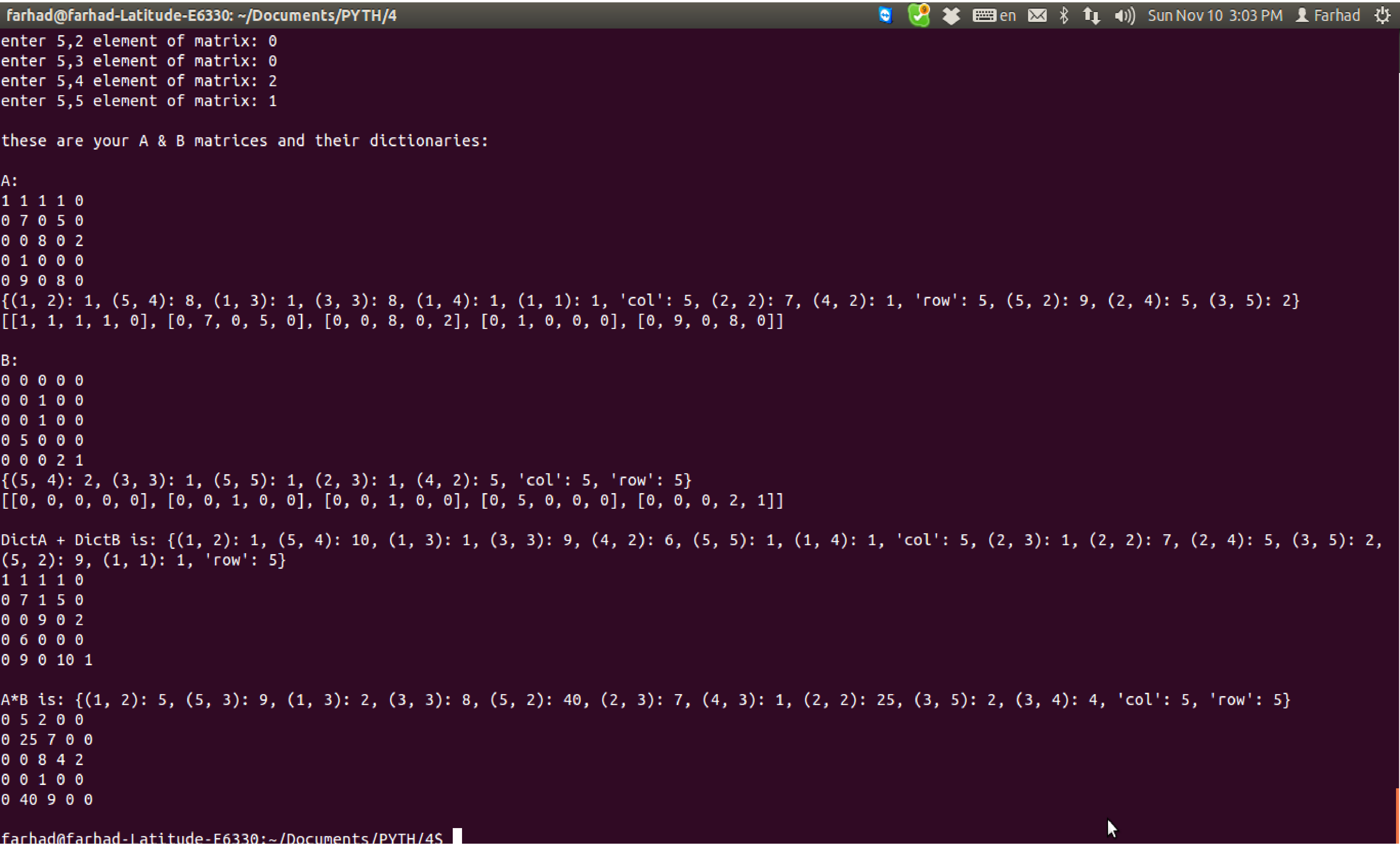
```
farhad@farhad-Latitude-E6330:~/Documents/PYTH/4$ libreoffice --writer
farhad@farhad-Latitude-E6330:~/Documents/PYTH/4$ python framezanghorbani13_hw4.py

these are your A & B matrices and their dictionaries:

A:
-1 1 0
0 0 1
7 0 9
{(1, 2): 1, (3, 3): 9, (3, 1): 7, 'col': 3, (2, 3): 1, (1, 1): -1, 'row': 3}
[[-1, 1, 0], [0, 0, 1], [7, 0, 9]]

B:
1 0 0
1 0 0
1 0 1
{(3, 3): 1, (3, 1): 1, (2, 1): 1, (1, 1): 1, 'col': 3, 'row': 3}
[[1, 0, 0], [1, 0, 0], [1, 0, 1]]

DictA + DictB is: {(1, 2): 1, (3, 3): 10, (3, 1): 8, (2, 1): 1, (2, 3): 1, 'col': 3, 'row': 3}
0 1 0
1 0 1
8 0 10

A*B is: {(3, 3): 9, (3, 1): 16, (2, 1): 1, (2, 3): 1, 'col': 3, 'row': 3}
0 0 0
1 0 1
16 0 9
```

For 2x3 and 3x3 matrices:



```
these are your A & B matrices and their dictionaries:

A:
-1 1 0
0 0 1
{(1, 2): 1, 'col': 3, (2, 3): 1, (1, 1): -1, 'row': 2}
[[-1, 1, 0], [0, 0, 1]]

B:
1 0 0
1 0 0
1 0 1
{(3, 3): 1, (3, 1): 1, (2, 1): 1, (1, 1): 1, 'col': 3, 'row': 3}
[[1, 0, 0], [1, 0, 0], [1, 0, 1]]


for addition two matrices should have same dimentions!
DictA + DictB is: None

A*B is: {(2, 3): 1, 'col': 3, (2, 1): 1, 'row': 2}
0 0 0
1 0 1

farhad@farhad-Latitude-E6330:~/Documents/PYTH/4$ 
```

And in next page you can see the result for 5x5 matrices:



```
enter 5,2 element of matrix: 0
enter 5,3 element of matrix: 0
enter 5,4 element of matrix: 2
enter 5,5 element of matrix: 1

these are your A & B matrices and their dictionaries:

A:
1 1 1 1 0
0 7 0 5 0
0 0 8 0 2
0 1 0 0 0
0 9 0 8 0
{(1, 2): 1, (5, 4): 8, (1, 3): 1, (3, 3): 8, (1, 4): 1, (1, 1): 1, 'col': 5, (2, 2): 7, (4, 2): 1, 'row': 5, (5, 2): 9, (2, 4): 5, (3, 5): 2}
[[1, 1, 1, 1, 0], [0, 7, 0, 5, 0], [0, 0, 8, 0, 2], [0, 1, 0, 0, 0], [0, 9, 0, 8, 0]]

B:
0 0 0 0 0
0 0 1 0 0
0 0 1 0 0
0 5 0 0 0
0 0 0 2 1
{(5, 4): 2, (3, 3): 1, (5, 5): 1, (2, 3): 1, (4, 2): 5, 'col': 5, 'row': 5}
[[0, 0, 0, 0, 0], [0, 0, 1, 0, 0], [0, 0, 1, 0, 0], [0, 5, 0, 0, 0], [0, 0, 0, 2, 1]]

DictA + DictB is: {(1, 2): 1, (5, 4): 10, (1, 3): 1, (3, 3): 9, (4, 2): 6, (5, 5): 1, (1, 4): 1, 'col': 5, (2, 3): 1, (2, 2): 7, (2, 4): 5, (3, 5): 2,
(5, 2): 9, (1, 1): 1, 'row': 5}
1 1 1 1 0
0 7 1 5 0
0 0 9 0 2
0 6 0 0 0
0 9 0 10 1

A*B is: {(1, 2): 5, (5, 3): 9, (1, 3): 2, (3, 3): 8, (5, 2): 40, (2, 3): 7, (4, 3): 1, (2, 2): 25, (3, 5): 2, (3, 4): 4, 'col': 5, 'row': 5}
0 5 2 0 0
0 25 7 0 0
0 0 8 4 2
0 0 1 0 0
0 40 9 0 0

farhad@farhad-Latitude-E6330:~/Documents/PYTH/4$
```