Akash Gupta. 2019DMB02

Program 1:

Listing out the n-factorial permuted matrices that form the determinant using basic functionality of nested loops and arrays.

Program 2:

Computing the Determinant of a square matrix by performing Row operations and Permutations to convert the Matrix into Upper Triangular form and calculating Determinant by multiplying the diagonal elements, while taking into account appropriate sign changes depending on the number of row permutations.

These programs have been written keeping in mind a general scenario for all kinds of square matrices while the demonstration has been done for a 3-order case. Additionally, the programs have been written using basic Nested loops functionality in Python.

```
# NAME: AKASH GUPTA
# SUBJECT: Operations Research
# ROLL NO: 2019DMB02

# Program 1

# This program has been written using in-built Python functionality and no external libraries
# have been used. Most of the functions that perform permutations and matrix manipulations
# are written using nested loops. Even though the demonstration is on a 3-order matrix, with
# different matrix inputs this program can perform for larger sizes of square matrices as well.

# A Program to list out all the possible n-factorial permutations of matrices containing only
# one element in each row and column. These are precisely the permuted matrices that form the
# Determinant. This program attempts to find all possible permutations and then outputs only
# the relevant ones for our purpose - that is the n-factorial permuted matrices that form
# the Determinant. Each such matrix has been listed out.


import random
A = [[1,5,3]
    ,[8,12,7]
    ,[3,1,4]]

print("NAME: AKASH GUPTA")
print()
print("ROLL NO: 2019DMB02")
print()
print("OR project: Program to list out n-factorial permuted matrices")
print()

print("The original matrix is shown below : ")
print()
prettyprint(A)

#-------------------------------------------------------------------------------#
# A simple print formatting function that prints out matrices in a clear manner
```

```
def prettyprint(A):
    for a in range(len(A)):
        for b in range(len(A[0])):
            print(A[a][b], end = "\t")
        print()

zero = [[0 for x in range(len(A))] for y in range(len(A))]
permut12 = []
per = []

#-------------------------------------------------------------------------------------------#
# The nested for loops are essentially permuting the elements and forming permuted matrices

for i in range(len(zero)):
    zero = [[0 for x in range(len(A))] for y in range(len(A))]
    zero[0][i] = A[0][i]
    for j in range(1,len(zero)):
        for k in range(len(zero)):
            zero[j] = [0 for x1 in range(len(A))]
            zero[j][k] = A[j][k]
            for k1 in range(2,len(zero)):
                for k2 in range(len(zero)):
                    zero[k1] = [0 for x2 in range(len(A))]
                    zero[k1][k2] = A[k1][k2]
                    per.extend(zero)
                if k1 > 1:
                    break
        if j > 0:
            break

a = []
for i in range(0, 81, 3):
    a.append(per[i:i+3])

vals = []   # The variable that will store the n-factorial permuted matrices

#-------------------------------------------------------------------------------------------#
# This set of nested loops checks if there are two elements belonging in the same column
# or same row, if they are then they are not considered. These loops separate out those
# permuted matrices that have one element for each column and row.

for n in a:
    booli = True
    count = 0
    for r in range(len(n)):
        count = 0
        for t in range(len(n)):
            if n[t][r] == 0:
                count += 1
        if count != 2:
            booli = False
            break
```

```
    if booli == True:
        vals.append(n)

#-------------------------------------------------#
# Printing out the n-factorial permuted matrices #

print("---------------")
print()
print("The n-factorial permuted matrices are shown below : ")
print()

for v in range(len(vals)):
    prettyprint(vals[v])
    print()
```

```
NAME: AKASH GUPTA

ROLL NO: 2019DMB02

OR project: Program to list out n-factorial permuted matrices

The original matrix is shown below :

1   5   3
8   12  7
3   1   4
---------------

The n-factorial permuted matrices are shown below :

1   0   0
0   12  0
0   0   4

1   0   0
0   0   7
0   1   0

0   5   0
8   0   0
0   0   4

0   5   0
0   0   7
3   0   0

0   0   3
8   0   0
0   1   0

0   0   3
0   12  0
3   0   0
```

```
# NAME: AKASH GUPTA
# SUBJECT: Operations Research
# ROLL NO: 2019DMB02

# Program 2

# This is an another program in relation to the Determinant concepts discussed in the lectures.
# This program performs row operations and row permutations to bring a matrix to an upper
# triangular form and then computes the Determinant by multiplying the diagonal elements.
# Additionally, the number of row permutations are counted so as to account for appropriate
# sign changes in the Determinant value.


from sys import exit

A = [[1,5,3]
    ,[8,12,7]
    ,[3,1,4]]

rowEx = 0

print("NAME: AKASH GUPTA")
print()
print("ROLL NO: 2019DMB02")
print()
print("Program to find the Determinant by matrix manipulations")
print()

#-------------------------------------------------------------------#
# A simple print function for printing matrices in a clear manner

def prettyprint(A):
    for a in range(len(A)):
        for b in range(len(A[0])):
            print(A[a][b], end = "\t")
        print()

#-----------------------------------------------------------------------------#
# These set of nested loops are performing row operations so as to bring the matrix in an
# upper triangular form so that Determinant can be computed by multiplying diagonal elements

for i in range(len(A)):
    if A[i][i] == 0:
        for i2 in range(i+1, len(A)):
            if A[i2][i2] != 0:
                for k in range(len(A[0])):
                    A[i][k],A[i2][k] = A[i2][k],A[i][k]
                rowEx += 1
                break
            else:
                exit(0)

    for j in range(i+1, len(A)):
```

```
        factor = A[j][i]/A[i][i]
        for k2 in range(len(A[0])):
            A[j][k2] = A[j][k2] - A[i][k2]*factor

prettyprint(A)
determinant = 1

for d in range(len(A)):
    determinant *= A[d][d]

#-------------------------------------------------------------------------------#
# rowEx is essentially a variable that counts the number of row permutations done on the matrix
# if the count of row permutations is odd, then the determinant will undergo a sign change
# which is what the if condition is doing

if rowEx%2 != 0:
    determinant = -determinant
print(determinant)
```

```
NAME: AKASH GUPTA

ROLL NO: 2019DMB02

Program to find the Determinant by matrix manipulations

1   5   3
0.0 -28.0   -17.0
0.0 0.0 3.5
-98.0
```