DSA ASSIGNMENT 3

1. Write a program to print the sum of the diagonal element of the M\*N square matrix.

*def* printDiagonalSums(*mat*, *n*):

    principal = 0

    for i in range(0, n):

        principal += mat[i][i]

    print("Principal Diagonal:", principal)

a = [[ 1, 2, 3, 4 ],

     [ 5, 6, 7, 8 ],

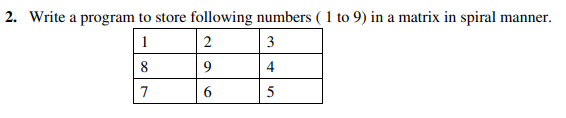
     [ 1, 2, 3, 4 ],

     [ 5, 6, 7, 8 ]]

print(a)

printDiagonalSums(a, 4)

OUTPUT:



*def* spiralOrder(*m*, *n*, *arr*):

    k, l, arrc=0,0,-1

    ScanArr = [[0 for i in range(m)] for i in range(n)]

    while(k<m and l<n):

        for i in range(l,n):

            arrc+=1

            ScanArr[k][i]=arr[arrc]

        k+=1

        for i in range(k,m):

            arrc+=1

            ScanArr[i][n-1]=arr[arrc]

        n-=1

        if (k<m):

            for i in range(n-1, l-1, -1):

                arrc+=1

                ScanArr[m-1][i]= arr[arrc]

            m-=1

        if(l<n):

            for i in range(m-1, k-1, -1):

                arrc+=1

                ScanArr[i][l]=arr[arrc]

            l+=1

    return ScanArr

arr=[1,2,3,4,5,6,7,8,9]

r=3

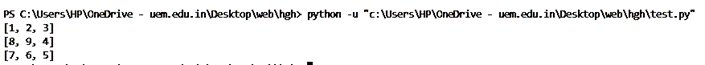
c=3

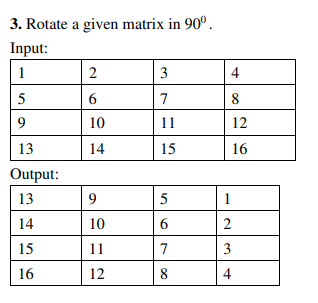
spiralOrder(r, c, arr)

for i in range(0,r):

    print(spiralOrder(r, c, arr)[i])

OUTPUT:





*def* rotate(*arr*):

    r= len(arr)

    c= len(arr[0])

    ar, ac= c, r

    arrRotated = [[0 for i in range(ar)] for i in range(ac)]

    for i in range(0, ac):

        for j in range(0, ar):

            arrRotated[j][i]=arr[r-i-1][j]

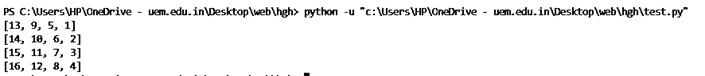
    return arrRotated

arr=[[1,2,3,4], [5,6,7,8], [9,10,11,12], [13,14,15,16]]

rotate(arr)

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

    print(rotate(arr)[i])

 OUTPUT:

4. Write a program to check whether a matrix is sparse or not.

a = [[4, 0, 0],[0, 5, 0],[0, 0, 6]];

count = 0;

rows = len(a);

cols = len(a[0]);

size = rows \* cols;

for i in range(0, rows):

for j in range(0, cols):

if(a[i][j] == 0):

count = count + 1;

if(count > (size/2)):

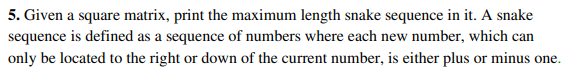
print("Given matrix is a sparse matrix");

else:

print("Given matrix is not a sparse matrix");

OUTPUT:





def snakesequence(S, m, n):

sequence = {}

DP = [[1 for x in range(m+1)] for x in range(n+1)]

a, b, maximum = 0, 0, 0

position = [0, 0]

for i in range(0, n+1):

for j in range(0, m+1):

a, b = 0, 0

p = "initial"

if(i > 0 and abs(S[i][j] - S[i-1][j]) == 1):

a = DP[i-1][j]

if(j > 0 and abs(S[i][j] - S[i][j-1]) == 1):

b = DP[i][j-1]

if a != 0 and a >= b:

p = str(i-1) + " " + str(j)

elif b != 0:

p = str(i) + " " + str(j-1)

q = str(i) + " " + str(j)

sequence[q] = p

DP[i][j] = DP[i][j] + max(a, b)

if DP[i][j] >= maximum:

maximum = DP[i][j]

position[0] = i

position[1] = j

snakeValues = []

snakePositions = []

snakeValues.append(S[position[0]][position[1]])

check = 'found'

str\_next = str(position[0]) + " " + str(position[1])

findingIndices = sequence[str\_next].split()

while(check == 'found'):

if sequence[str\_next] == 'initial':

snakePositions.insert(0, str\_next)

check = 'end'

continue

findingIndices = sequence[str\_next].split()

g = int(findingIndices[0])

h = int(findingIndices[1])

snakeValues.insert(0, S[g][h])

snake\_position = str(g) + " " + str(h)

snakePositions.insert(0, str\_next)

str\_next = sequence[str\_next]

return [snakeValues, snakePositions]

S = [[9, 6, 5, 2],

[8, 7, 6, 5],

[7, 3, 1, 6],

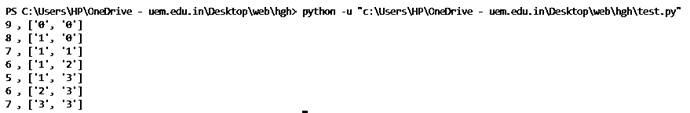
[1, 1, 10, 7]]

m = 3

n = 3

seq = snakesequence(S, m, n)

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

 print(seq[0][i], ",", seq[1][i].split())

#include <stdio.h>

int main(){

int Matrix[3][3] = {{ 1, 2, 3 },{ 4, 5, 6 },{ 7, 8, 9 }};

int i, j, n1, n2, temp;

printf("Matrix before row exchange:\n");

for (i = 0; i < 3; ++i) {

for (j = 0; j < 3; ++j)

printf(" %d", Matrix[i][j]);

printf("\n");}

printf("Enter two row numbers to be exchanged:");

scanf("%d %d", &n1, &n2);

for (i = 0; i < 3; ++i) {

temp = Matrix[n1 - 1][i];

Matrix[n1 - 1][i] = Matrix[n2 - 1][i];

Matrix[n2 - 1][i] = temp;}

printf("Matrix after row exchange:\n");

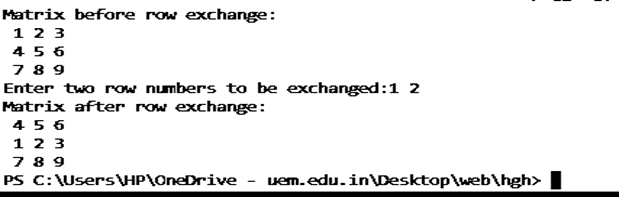
for (i = 0; i < 3; ++i) {

for (j = 0; j < 3; ++j)

printf(" %d", Matrix[i][j]);

printf("\n");}

return 0;}





def reverse(string):

string = string[::-1]

return string

s = input("Enter a string: ")

print("The original string is : ", end="")

print(s)

print("The reversed string is : ", end="")

print(reverse(s))





def reverse(string):

string = string[::-1]

return string

s = input("Enter a string: ")

if(s==reverse(s)):

print("palindrome")

else:

print("not palindrome")





def fxn(stng):

oupt = stng[0]

for i in range(1, len(stng)):

if stng[i-1] == ' ':

oupt += stng[i]

oupt = oupt.upper()

return oupt

inpt1 = input("Enter string to abbreviate: ")

print(fxn(inpt1))





lst = ['uem', 'kolkata', 'is', 'a', 'good', 'college']

lst.sort()

print(lst)

print("Enter an element to be search: ")

elem = (input())

flag =1;

for i in range(5):

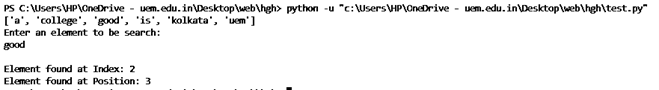
if elem == lst[i]:

print("\nElement found at Index:", i)

print("Element found at Position:", i+1)

flag+=1

if flag ==1:

 print("Element not found")