מגיש: אביחי חדד(209286665)

**תרגיל 1:** קוד:

# 1-

def is\_row\_equal(matrix, row\_num):

""" Function is\_row\_equal gets a matrix and number

and returns if the sum of all number in that row

is equal to the received num.

"""

sum\_of\_row = 0

# running number of cols of that row

# [number of elements in list].

for i in range(len(matrix[row\_num])):

sum\_of\_row += matrix[row\_num][i]

return sum\_of\_row == row\_num

def is\_diagonal\_positive(matrix):

""" Function is\_diagonal\_positive gets a matrix

and returns if all nuumbers in the diagonal are positive.

"""

# running len(rows) times.

for i in range(len(matrix)):

# checking the possibility that it's not a square matrix.

if len(matrix[i]) < i + 1:

return True

if matrix[i][i] < 0:

return False

return True

def print\_matrix(matrix):

""" Function print\_matrix get a matrix

and prints it with space of 4 to the left between elements.

"""

# run on rows

for i in range(len(matrix)):

# runs on col of each row

for j in range(len(matrix[0])):

print("%4d" % matrix[i][j], end=" ")

print()

def is\_mat\_ok(matrix):

""" Function is\_mat\_ok gets a matrix

and retruns if the matrix is ok by calling

"is\_row\_equal" and "is\_diagonal\_positive" functions.

"""

# running on each row of matrix.

for i in range(len(matrix)):

if not is\_row\_equal(matrix, i):

return False

return is\_diagonal\_positive(matrix)

def main():

""" Function main has hard-coded matrices and prints them

and print if the matrices are ok(according to requirements).

"""

matrix\_a = [[31, -15, 0, -12, -4],

[1, 1, -3, 2, 0],

[12, -2, 4, -23, 11],

[5, 0, 3, 2, -7],

[1, 1, 0, 1, 1]]

matrix\_b = [[1, 2, 3, 4, 5],

[6, 7, 8, 9, 0],

[1, 2, 3, 4, 5],

[6, 7, 8, 9, 0],

[1, 2, 3, 4, 5]]

print\_matrix(matrix\_a)

print("Matrix A is", "ok" if is\_mat\_ok(matrix\_a) else "not ok")

print()

print\_matrix(matrix\_b)

print("Matrix B is", "ok" if is\_mat\_ok(matrix\_b) else "not ok")

main()

|  |
| --- |
|  |

**פלט:**

**תרגיל 2:** קוד:

# 2-

def get\_matrix\_from\_user(dim):

""" Function get\_matrix\_from\_user gets a dimension of a matrix

and return a user inputted matrix.

"""

matrix = []

row\_list = []

print("Enter the entries rowwise:")

# running "dim" times, for rows.

for i in range(dim):

# running "dim" times, for cols.

for j in range(dim):

row\_list.append(int(input()))

# after getting all element of a row, appending it to the matrix.

matrix.append(row\_list)

row\_list = []

return matrix

def print\_matrix(matrix):

""" Function print\_matrix get a matrix

and prints it with space of 4 to the left between elements.

"""

# run on rows

for i in range(len(matrix)):

# runs on col of each row

for j in range(len(matrix[0])):

print("%4d" % matrix[i][j], end=" ")

print()

def is\_mat\_perfect(matrix, dim):

""" Function is\_mat\_perfect gets a matrix and the dimension of it

and returns whether the matrix is perfect or not.

"""

# two list of counter for row, and col.

row\_counter\_list = [0] \* dim

col\_counter\_list = [0] \* dim

# running on rows.

for i in range(dim):

# running on cols.

for j in range(dim):

# checking if element is in bounds.

if (0 < matrix[i][j] <= dim) and (0 < matrix[j][i] <= dim):

# incrementing the right slot that represents the number,

# counter\_list[0] represents number of 1's

row\_counter\_list[matrix[i][j] - 1] += 1

col\_counter\_list[matrix[j][i] - 1] += 1

# if not in bounds.

else:

return False

# checking if the row/cols have more than one representative of a digit.

if max(row\_counter\_list) > 1 or max(col\_counter\_list) > 1:

return False

row\_counter\_list = [0] \* dim

col\_counter\_list = [0] \* dim

return True

def main():

""" Function main gets a dimension and matrix from user

and prints if the matrix is perfect or not(according to requirements).

"""

# an endlees loop until dimension is 0.

while True:

dimension = int(input("Enter the matrix dimension: "))

if dimension == 0:

print("Finish")

break

matrix = get\_matrix\_from\_user(dimension)

print\_matrix(matrix)

print("The Matrix is", "perfect" if is\_mat\_perfect(matrix, dimension) \

else "not perfect")

|  |
| --- |
|  |

main()

פלט: