""" Python3 program to solve N Queen Problem

using Branch or Bound """

N = 8

""" A utility function to print solution """

def printSolution(board):

    for i in range(N):

        for j in range(N):

            print(board[i][j], end = " ")

        print()

""" A Optimized function to check if

a queen can be placed on board[row][col] """

def isSafe(row, col, slashCode, backslashCode,

           rowLookup, slashCodeLookup,

                       backslashCodeLookup):

    if (slashCodeLookup[slashCode[row][col]] or

        backslashCodeLookup[backslashCode[row][col]] or

        rowLookup[row]):

        return False

    return True

""" A recursive utility function

   to solve N Queen problem """

def solveNQueensUtil(board, col, slashCode, backslashCode,

                     rowLookup, slashCodeLookup,

                     backslashCodeLookup):

    """ base case: If all queens are

       placed then return True """

    if(col >= N):

        return True

    for i in range(N):

        if(isSafe(i, col, slashCode, backslashCode,

                  rowLookup, slashCodeLookup,

                  backslashCodeLookup)):

            """ Place this queen in board[i][col] """

            board[i][col] = 1

            rowLookup[i] = True

            slashCodeLookup[slashCode[i][col]] = True

            backslashCodeLookup[backslashCode[i][col]] = True

            """ recur to place rest of the queens """

            if(solveNQueensUtil(board, col + 1,

                                slashCode, backslashCode,

                                rowLookup, slashCodeLookup,

                                backslashCodeLookup)):

                return True

            """ If placing queen in board[i][col]

            doesn't lead to a solution,then backtrack """

            """ Remove queen from board[i][col] """

            board[i][col] = 0

            rowLookup[i] = False

            slashCodeLookup[slashCode[i][col]] = False

            backslashCodeLookup[backslashCode[i][col]] = False

    """ If queen can not be place in any row in

    this column col then return False """

    return False

""" This function solves the N Queen problem using

Branch or Bound. It mainly uses solveNQueensUtil()to

solve the problem. It returns False if queens

cannot be placed,otherwise return True or

prints placement of queens in the form of 1s.

Please note that there may be more than one

solutions,this function prints one of the

feasible solutions."""

def solveNQueens():

    board = [[0 for i in range(N)]

                for j in range(N)]

    # helper matrices

    slashCode = [[0 for i in range(N)]

                    for j in range(N)]

    backslashCode = [[0 for i in range(N)]

                        for j in range(N)]

    # arrays to tell us which rows are occupied

    rowLookup = [False] \* N

    # keep two arrays to tell us

    # which diagonals are occupied

    x = 2 \* N - 1

    slashCodeLookup = [False] \* x

    backslashCodeLookup = [False] \* x

    # initialize helper matrices

    for rr in range(N):

        for cc in range(N):

            slashCode[rr][cc] = rr + cc

            backslashCode[rr][cc] = rr - cc + 7

    if(solveNQueensUtil(board, 0, slashCode, backslashCode,

                        rowLookup, slashCodeLookup,

                        backslashCodeLookup) == False):

        print("Solution does not exist")

        return False

    # solution found

    printSolution(board)

    return True

""" Python3 program to solve N Queen Problem using backtracking """

B= 4

""" ld is an array where its indices indicate row-col+N-1

(N-1) is for shifting the difference to store negative

indices """

ld = [0] \* 30

""" rd is an array where its indices indicate row+col

and used to check whether a queen can be placed on

right diagonal or not"""

rd = [0] \* 30

"""column array where its indices indicates column and

used to check whether a queen can be placed in that

    row or not"""

cl = [0] \* 30

""" A utility function to print solution """

def printSolutionB(board):

    for i in range(B):

        for j in range(B):

            print(board[i][j], end = " ")

        print()

""" A recursive utility function to solve N

Queen problem """

def solveNQUtilB(board, col):

    """ base case: If all queens are placed

        then return True """

    if (col >= B):

        return True

    """ Consider this column and try placing

        this queen in all rows one by one """

    for i in range(B):

        """ Check if the queen can be placed on board[i][col] """

        """ A check if a queen can be placed on board[row][col].

        We just need to check ld[row-col+n-1] and rd[row+coln]

        where ld and rd are for left and right diagonal respectively"""

        if ((ld[i - col + B - 1] != 1 and

             rd[i + col] != 1) and cl[i] != 1):

            """ Place this queen in board[i][col] """

            board[i][col] = 1

            ld[i - col + B - 1] = rd[i + col] = cl[i] = 1

            """ recur to place rest of the queens """

            if (solveNQUtilB(board, col + 1)):

                return True

            """ If placing queen in board[i][col]

            doesn't lead to a solution,

            then remove queen from board[i][col] """

            board[i][col] = 0 # BACKTRACK

            ld[i - col + B - 1] = rd[i + col] = cl[i] = 0

            """ If the queen cannot be placed in

            any row in this column col then return False """

    return False

""" This function solves the N Queen problem using

Backtracking. It mainly uses solveNQUtil() to

solve the problem. It returns False if queens

cannot be placed, otherwise, return True and

prints placement of queens in the form of 1s.

Please note that there may be more than one

solutions, this function prints one of the

feasible solutions."""

def solveNQB():

    board = [[0, 0, 0, 0],

             [0, 0, 0, 0],

             [0, 0, 0, 0],

             [0, 0, 0, 0]]

    if (solveNQUtilB(board, 0) == False):

        printf("Solution does not exist")

        return False

    printSolutionB(board)

    return True

# Driver Code

flag=1

while flag==1:

    print("1. solve N Queen Problem using Branch or Bound \n 2. Queen Problem using backtracking \n 3. Exit\n")

    ch=int(input("Enter your Choice (from 1 to 3) :"))

    if ch==1:

        print("solution of  N Queen Problem using Branch or Bound")

        solveNQueens()   # function calling

        a = input("Do you want to continue (y/n) :")

        if a == "y":

            flag = 1

        else:

            flag = 0

            print("Thanks for using this program!")

    elif ch==2:

        print("solution of Queen Problem using backtracking")

        solveNQB()

        a = input("Do you want to continue (y/n) :")

        if a == "y":

            flag = 1

        else:

            flag = 0

            print("Thanks for using this program!")

    elif ch==3:

        flag=0

        print("Thanks for using this program!")

    else:

        print("!!Wrong Choice!! ")

        a=input("Do you want to continue (y/n) :")

        if a=="y":

            flag=1

        else:

            flag=0

            print("Thanks for using this program!")