# -\*- coding: utf-8 -\*-

**"""5.Implement cryptarithmetic(Branch,Bound amd backtracking).ipynb**

Automatically generated by Colaboratory.

Original file is located at

https://colab.research.google.com/drive/1ZVJehZEeu2WmtDPKC2znwf5KeiLl8p\_y

"""

N=int(input("Enter number of queens :"))

Board=[['\_' for x in range(N)] for y in range(N)]

print(Board)

def printBoard(Board):

for i in Board:

for j in i:

print(j,end =" ")

print(' ')

def isSafe(Board,row,col):

for i in range(col):

if(Board[row][i]=='Q'):

return False

for i,j in zip(range(row,-1,-1),range(col,-1,-1)):

if(Board[i][j]=='Q'):

return False

for i,j in zip(range(row,N,1),range(col,-1,-1)):

if(Board[i][j]=='Q'):

return False

return True

def SolveQueen(Board,col):

if(col>=N):

return True

for i in range(N):

if(isSafe(Board,i,col)==True):

Board[i][col]='Q'

printBoard(Board)

print(' ')

if(SolveQueen(Board,col+1)==True):

return True

Board[i][col]='\_'

print ("\n....BackTracking here...")

return False

if SolveQueen(Board,0)==False:

print ("\nSolution not exist")

else:

print ("\nFinal Solution")

printBoard(Board)

Output:

Enter number of queens :4

[['\_', '\_', '\_', '\_'], ['\_', '\_', '\_', '\_'], ['\_', '\_', '\_', '\_'], ['\_', '\_', '\_', '\_']]

Q \_ \_ \_

\_ \_ \_ \_

\_ \_ \_ \_

\_ \_ \_ \_

Q \_ \_ \_

\_ \_ \_ \_

\_ Q \_ \_

\_ \_ \_ \_

....BackTracking here...

Q \_ \_ \_

\_ \_ \_ \_

\_ \_ \_ \_

\_ Q \_ \_

Q \_ \_ \_

\_ \_ Q \_

\_ \_ \_ \_

\_ Q \_ \_

....BackTracking here...

....BackTracking here...

....BackTracking here...

\_ \_ \_ \_

Q \_ \_ \_

\_ \_ \_ \_

\_ \_ \_ \_

\_ \_ \_ \_

Q \_ \_ \_

\_ \_ \_ \_

\_ Q \_ \_

\_ \_ Q \_

Q \_ \_ \_

\_ \_ \_ \_

\_ Q \_ \_

\_ \_ Q \_

Q \_ \_ \_

\_ \_ \_ Q

\_ Q \_ \_

Final Solution

\_ \_ Q \_

Q \_ \_ \_

\_ \_ \_ Q

\_ Q \_ \_