Neha Dattatray Bhoite

Assignment No 4

1)N Queen Problem

```
# Taking number of queens as input
from user
print ("Enter the number of queens")
N = int(input())
# here we create a chessboard
# NxN matrix with all elements set to 0
board = [[0]*N for _ in range(N)]
def attack(i, j):
    #checking vertically and horizontally
    for k in range(0,N):
        if board[i][k]==1 or board[k][j]==1:
            return True
    #checking diagonally
    for k in range(0,N):
        for l in range(0,N):
            if (k+l==i+j) or (k-l==i-j):
                if board[k][l]==1:
                    return True
    return False
def N_queens(n):
    if n==0:
        return True
    for i in range(0,N):
        for j in range(0,N):
            if (not(attack(i,j))) and
(board[i][j]!=1):
                board[i][j] = 1
                if N_queens(n-1)==True:
                    return True
                board[i][j] = 0
    return False
N_queens(N)
for i in board:
    print (i)
```

Output-

```
Enter the number of queens
4
[0, 1, 0, 0]
[0, 0, 0, 1]
[1, 0, 0, 0]
[0, 0, 1, 0]

=== Code Execution Successful ===
```

2) graph coloring problem

```
def isSafe(graph, color):
  # check for every edge
   for i in range(4):
      for j in range(i + 1, 4):
         if (graph[i][j] and color[j] ==
color[i]):
            return False
   return True

def graphColoring(graph, m, i, color):

   # if current index reached end
   if (i == 4):

      # if coloring is safe
      if (isSafe(graph, color)):

         # Print the solution
         printSolution(color)
         return True
      return False

   # Assign each color from 1 to m
   for j in range(1, m + 1):
      color[i] = j

      # Recur of the rest vertices
      if (graphColoring(graph, m, i + 1,
color)):
         return True
      color[i] = 0
   return False

# /* A utility function to print solution */

 def printSolution(color):
   print("Solution Exists:" " Following
are the assigned colors ")
for i in range(4):
      print(color[i], end=" ")

 # Driver code
if __name__ == '__main__':

   # /* Create following graph and
   # test whether it is 3 colorable
   # (3)---(2)
   # | / |
   # | / |
   # | / |
   # (0)---(1)
   # */
   graph = [
      [0, 1, 1, 1],
      [1, 0, 1, 0],
      [1, 1, 0, 1],
      [1, 0, 1, 0],
   ]
   m = 3  # Number of colors

   # Initialize all color values as 0.
   # This initialization is needed
   # correct functioning of isSafe()
   color = [0 for i in range(4)]

   # Function call
   if (not graphColoring(graph, m, 0,
color)):
      print("Solution does not exist")
```

Output-

```
Solution Exists: Following are the
assigned colors
1 2 3 2
=== Code Execution Successful ===
```