Akshat Agarwal
RA1911003010646

Artificial Intelligence
Lab - 1

Aim : To implement a toy problem, i.e. 8 queens problem

Algorithm :

1. Start from the leftmost column
2. If all queens are placed return true
3. Try all rows in the current column. Do following for every tried row.

   a) If the queen can be placed safely in this row then mark this [row, column] as part of the solution and recursively check if placing queen here leads to a solution.

   b) If placing the queen in [row, column] leads to a solution then return true.

   c) If placing queen doesn't lead to a solution then unmark this [row, column] (Backtrack) and go to step (a) to try other rows.

4. If all rows have been tried and nothing worked, return false to trigger backtracking.

Code :

```
print ("Enter the number of queens")
N = int(input())

board = [[0]*N for _ in range(N)]

def attack(i, j):
```

```
        for k in range(0,N):
            if board[i][k]==1 or board[k][j]==1:
                return True

        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
8
[1, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 1, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 1]
[0, 0, 0, 0, 0, 1, 0, 0]
[0, 0, 1, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 1, 0]
[0, 1, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 1, 0, 0, 0, 0]
```

RA1911003010646
Akshat Agarwal
STUDY BUDDIES

AI LAB-1

8 Queen's Prob.

Algorithm:

1. Start from the leftmost column
2. If all queens are placed return true
3. Try all rows in the current column. Do following for every tried row
   a) if the queen can be placed safely in this row then mark this [row, column] as part of the solution & recursively check if placing queen here leads to a solution.
   b) if placing the queen in [row, column] leads to a solution then return true
   c) if placing queen doesn't lead to a solution then unmark this [row, column] (Backtrack) and go to step(a) to try other rows
4. If all rows have been tried and nothing worked, return false to trigger backtracking.