Bushra dajam

Lab5

2110054

```python
import numpy as np
import copy
#Define empty grid (any size is okay)
grid=[[0,0,0,0,0,0,0,0],
      [0,0,0,0,0,0,0,0],
      [0,0,0,0,0,0,0,0],
      [0,0,0,0,0,0,0,0],
      [0,0,0,0,0,0,0,0],
      [0,0,0,0,0,0,0,0],
      [0,0,0,0,0,0,0,0],
      [0,0,0,0,0,0,0,0]]
#or
N=8
grid=np.zeros([N,N], dtype=int)
grid=grid.tolist()

def possible(grid,y,x): #is it possibl e to place a queen into y, x?

    l=len(grid) #how big is our grid?
    for i in range(l): #check for queens on row y
        if grid[y][i]==1: #if exist return false
            return False
    for i in range(l): #check for queens on column x
        if grid[i][x]==1: #if exists return o
            return False

    for i in range(l): #Loop through all rows
        for j in range(l): #and columns
            if grid[i][j]==1: #if there is a queen
                if abs (i - y) == abs(j - x): #and if there is another on a diagonal
                    return False #return false
    return True #if every check clears, we can return true
```

```python
def possible(grid,y,x): #is it possibl e to place a queen into y, x?

    l=len(grid) #how big is our grid?
    for i in range(l): #check for queens on row y
        if grid[y][i]==1: #if exist return false
            return False
    for i in range(l): #check for queens on column x
        if grid[i][x]==1: #if exists return o
            return False

    for i in range(l): #Loop through all rows
        for j in range(l): #and columns
            if grid[i][j]==1: #if there is a queen
                if abs (i - y) == abs(j - x): #and if there is another on a diagonal
                    return False #return false
    return True #if every check clears, we can return true

def solve(grid):

    l=len(grid)     #Length of our grid
    grid[7][0]=1
    for y in range(l):  #for every row
        for x in range(l):  #for every column
            if grid[y][x]==0:   # we can place if there is no queen in given position
                if possible(grid,y, x):     #if empty, check if we can place a queen
                    grid[y][x]=1    #if we can, then place it
                    solve (grid)    #pass grid for recursive solution
                    #if we end up here, means we searched through all children branches
                    if sum (sum(a) for a in grid)==l:   #if there are 8 queens
                        return grid     #we are successful so return
                    grid[y][x]=0     #remove the previous placed queen
    return grid     #means we searched the space, we can return our result

Solution = solve(copy.deepcopy(grid)) #get the solution
print(np.matrix(Solution)) #Print the solution
```

Output:

main.py

```
    l=len(grid) #how big is our grid?
    for i in range(l): #check for queens on row y
        if grid[y][i]==1: #if exist return false
            return False
    for i in range(l): #check for queens on column x
        if grid[i][x]==1: #if exists return o
            return False

    for i in range(l): #Loop through all rows
        for j in range(l): #and columns
            if grid[i][j]==1: #if there is a queen
                if abs (i - y) == abs(j - x): #and if there is another on a diagonal
                    return False #return false
```

input

```
[[0 0 1 0 0 0 0 0]
 [0 0 0 0 1 0 0 0]
 [0 1 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 1]
 [0 0 0 0 0 1 0 0]
 [0 0 0 1 0 0 0 0]
 [0 0 0 0 0 0 1 0]
 [1 0 0 0 0 0 0 0]]


...Program finished with exit code 0
Press ENTER to exit console.
```