

Bushra dajam

Lab5

2110054

```
1 import numpy as np
2 import copy
3 #Define empty grid (any size is okay)
4 grid=[[0,0,0,0,0,0,0,0],
5        [0,0,0,0,0,0,0,0],
6        [0,0,0,0,0,0,0,0],
7        [0,0,0,0,0,0,0,0],
8        [0,0,0,0,0,0,0,0],
9        [0,0,0,0,0,0,0,0],
10       [0,0,0,0,0,0,0,0],
11       [0,0,0,0,0,0,0,0]]
12
13 #or
14 N=8
15 grid=np.zeros([N,N], dtype=int)
16 grid=grid.tolist()
17
18 def possible(grid,y,x): #is it possible to place a queen into y, x?
19     l=len(grid) #how big is our grid?
20     for i in range(1): #check for queens on row y
21         if grid[y][i]==1: #if exist return false
22             return False
23     for i in range(1): #check for queens on column x
24         if grid[i][x]==1: #if exists return 0
25             return False
26
27     for i in range(1): #Loop through all rows
28         for j in range(1): #and columns
29             if grid[i][j]==1: #if there is a queen
30                 if abs(i - y) == abs(j - x): #and if there is another on a diagonal
31                     return False #return false
32     return True #if every check clears, we can return true
33
34 def solve(grid):
35     l=len(grid) #Length of our grid
36     grid[7][6]=1
37     for y in range(1): #for every row
38         for x in range(1): #for every column
39             if grid[y][x]==0: # we can place if there is no queen in given position
40                 if possible(grid,y, x): #if empty, check if we can place a queen
41                     grid[y][x]=1 #if we can, then place it
42                     solve (grid) #pass grid for recursive solution
43                     #if we end up here, means we searched through all children branches
44                     if sum(sum(a) for a in grid)==1: #if there are 8 queens
45                         return grid #we are successful so return
46                     grid[y][x]=0 #remove the previous placed queen
47                     #means we searched the space, we can return our result
48     return grid
49
50 Solution = solve(copy.deepcopy(grid)) #get the solution
51 print(np.matrix(Solution)) #Print the solution
```

Output:

OnlineGDB beta

online compiler and debugger for c/c++

code. compile. run. debug. share.

IDE

My Projects

Classroom new

Learn Programming

Programming Questions


Sign Up

Login

f

t

+ 1.6K



GOT AN OPINION?

SHARE AND GET REWARDS.

OkutenAP

Have fun taking surveys and get paid!

ADS VIA CARBON

About • FAQ • Blog • Terms of Use • Contact Us •

main.py

```
19 l=len(grid) #how big is our grid?
20 for i in range(1): #check for queens on row y
21     if grid[y][i]==1: #if exist return false
22         return False
23 for i in range(1): #check for queens on column x
24     if grid[i][x]==1: #if exists return o
25         return False
26
27 for i in range(1): #Loop through all rows
28     for j in range(1): #and columns
29         if grid[i][j]==1: #if there is a queen
30             if abs(i - y) == abs(j - x): #and if there is another on a diagonal
31                 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.
```