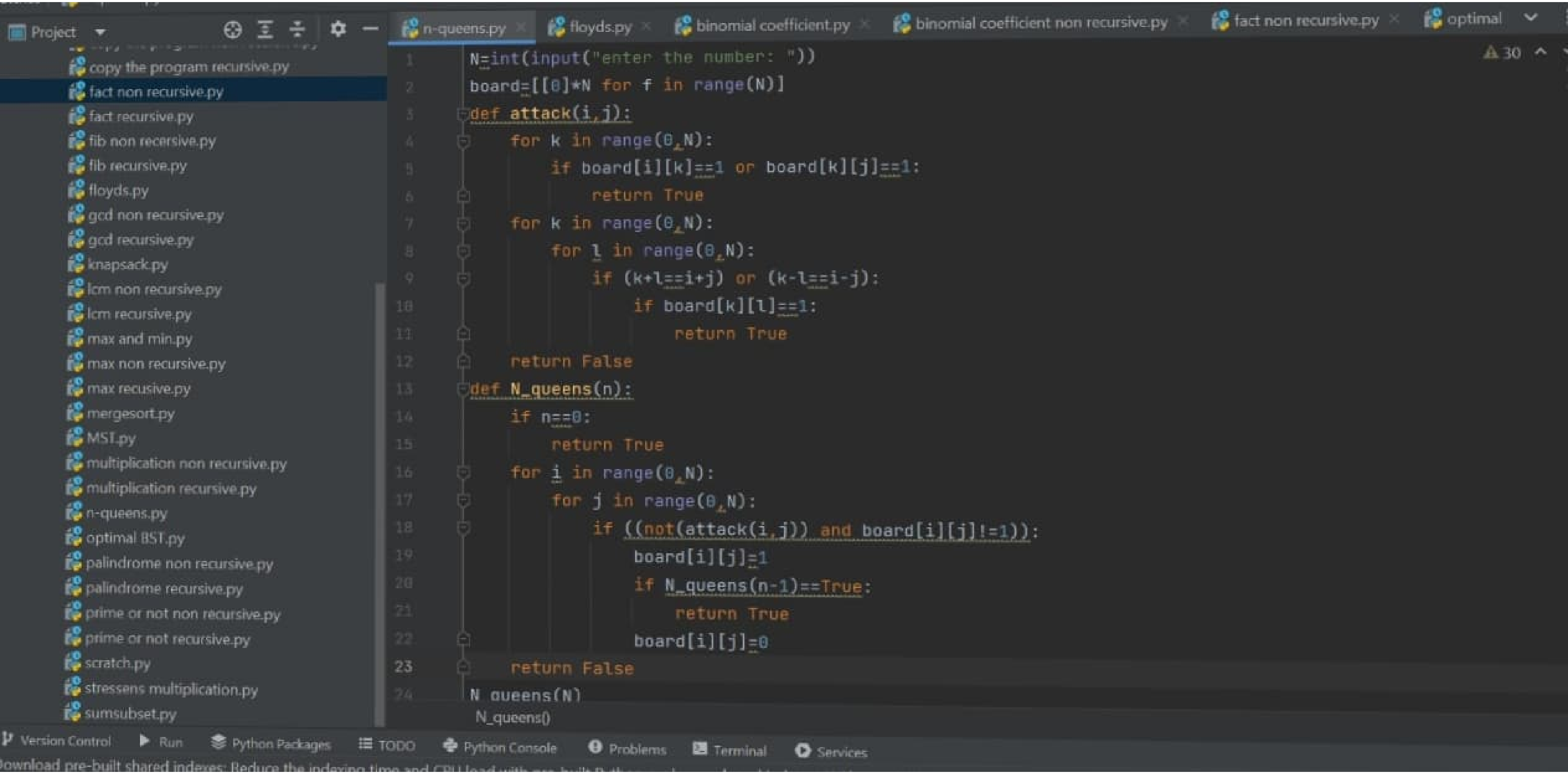
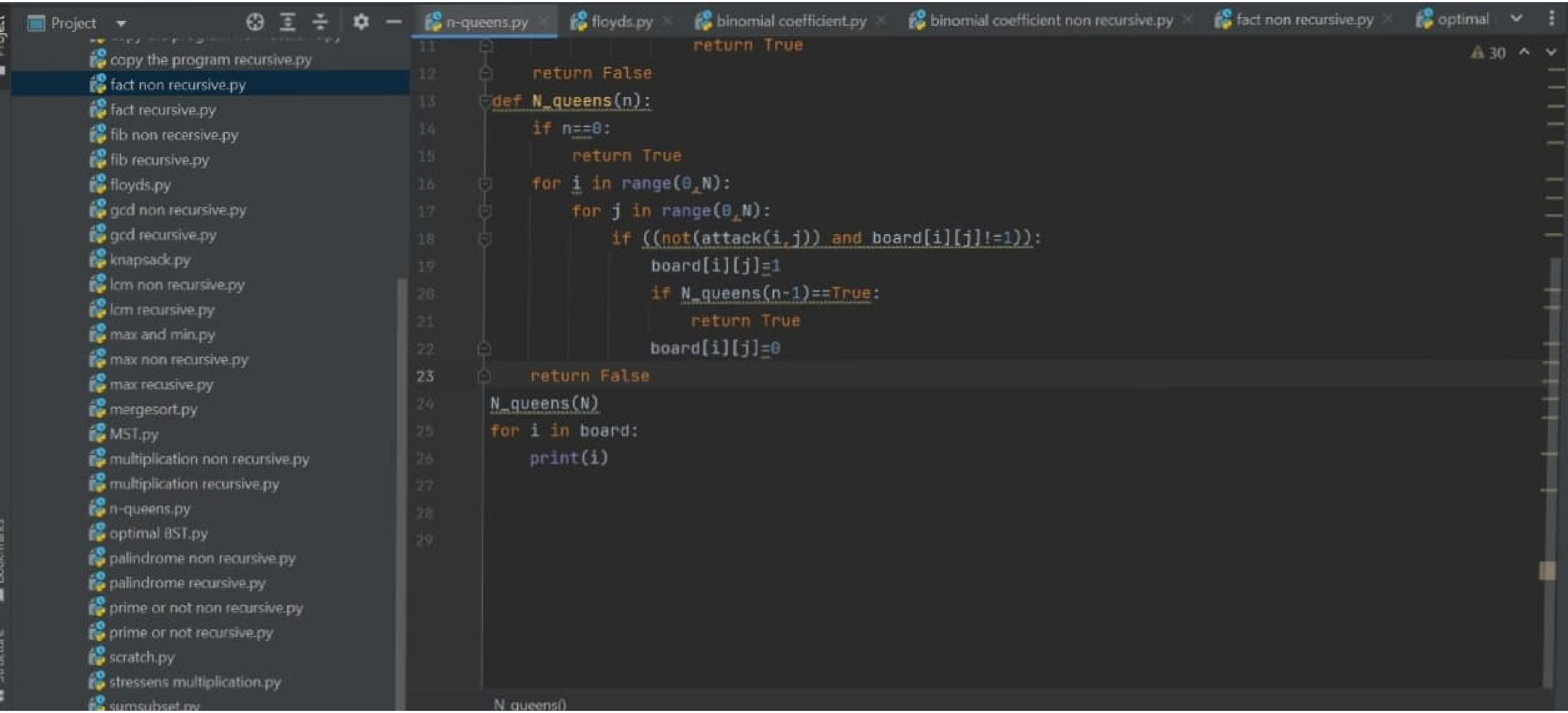
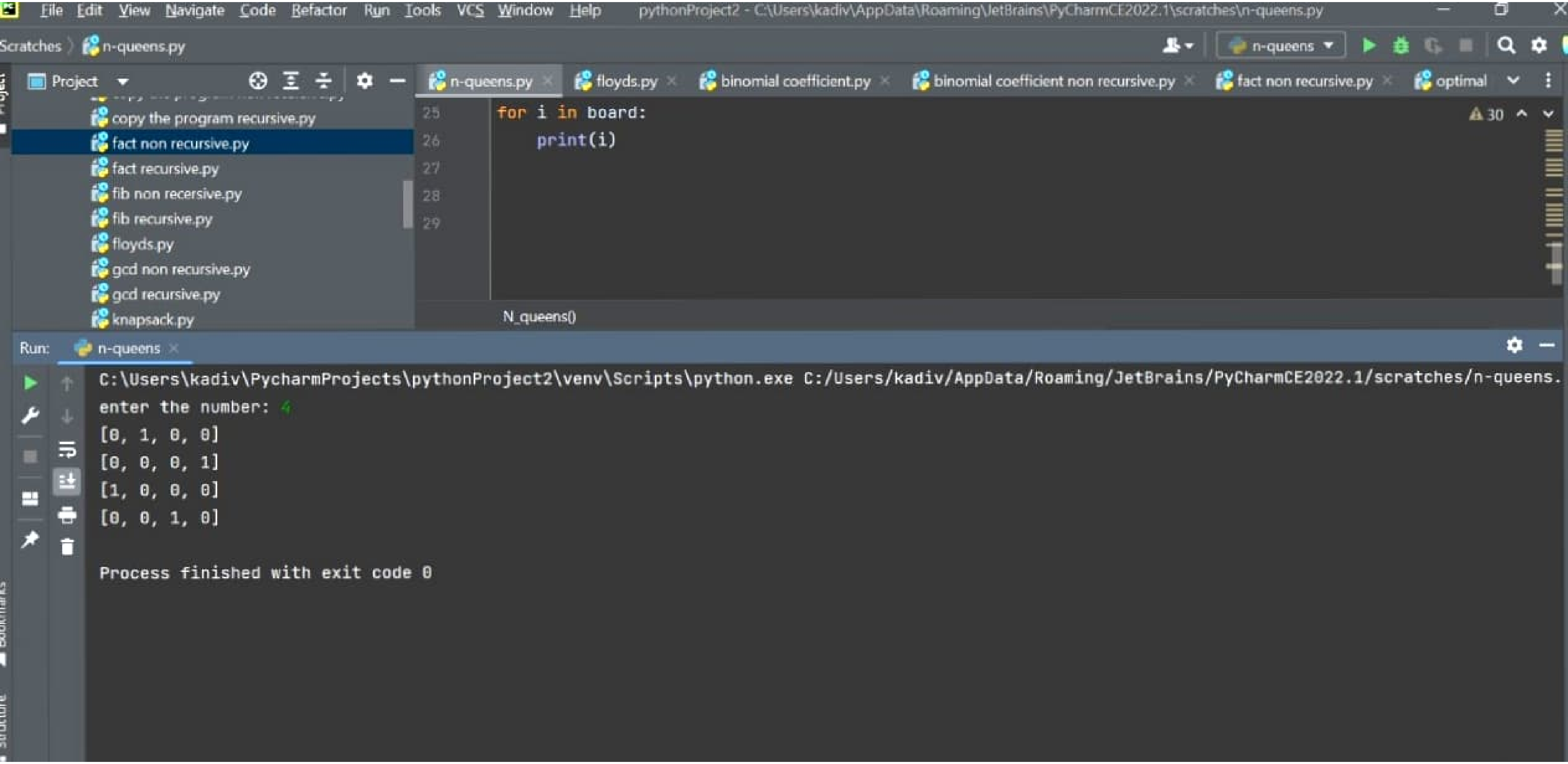

Project
n-queens.py
floyds.py
binomial coefficient.py
binomial coefficient non recursive.py
fact non recursive.py
optimal
30
copy the program recursive.py
fact non recursive.py
fact recursive.py
fib non recersive.py
fib recursive.py
floyds.py
gcd non recursive.py
gcd recursive.py
knapsack.py
lcm non recursive.py
lcm recursive.py
max and min.py
max non recursive.py
max recusive.py
mergesort.py
MST.py
multiplication non recursive.py
multiplication recursive.py
n-queens.py
optimal BST.py
palindrome non recursive.py
palindrome recursive.py
prime or not non recursive.py
prime or not recursive.py
scratch.py
stressens multiplication.py
sumsubset.py
N_queens()
Version Control
Run
Python Packages
TODO
Python Console
Problems
Terminal
Services
Download pre-built shared indexes: Reduce the indexing time and CPU load with pre-built P

```
N=int(input("enter the number: "))
board=[[0]*N for f 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)
```


Project
n-queens.py
floyds.py
binomial coefficient.py
binomial coefficient non recursive.py
fact non recursive.py
optimal
30
copy the program recursive.py
fact non recursive.py
fact recursive.py
fib non recersive.py
fib recursive.py
floyds.py
gcd non recursive.py
gcd recursive.py
knapsack.py
lcm non recursive.py
lcm recursive.py
max and min.py
max non recursive.py
max recusive.py
mergesort.py
MST.py
multiplication non recursive.py
multiplication recursive.py
n-queens.py
optimal BST.py
palindrome non recursive.py
palindrome recursive.py
prime or not non recursive.py
prime or not recursive.py
scratch.py
stressens multiplication.py
sumsubset.py
N_queens()

```
                    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)
```


File Edit View Navigate Code Refactor Run Tools VCS Window Help
pythonProject2 - C:\Users\kadiv\AppData\Roaming\JetBrains\PyCharmCE2022.1\scratches\n-queens.py
Scratches
n-queens.py
n-queens
Project
copy the program recursive.py
fact non recursive.py
fact recursive.py
fib non recersive.py
fib recursive.py
floyds.py
gcd non recursive.py
gcd recursive.py
knapsack.py
n-queens.py
floyds.py
binomial coefficient.py
binomial coefficient non recursive.py
fact non recursive.py
optimal
30
N_queens()
Run:
n-queens

```
for i in board:
    print(i)
```

```
C:\Users\kadiv\PycharmProjects\pythonProject2\venv\Scripts\python.exe C:/Users/kadiv/AppData/Roaming/JetBrains/PyCharmCE2022.1/scratches/n-queens.
enter the number: 4
[0, 1, 0, 0]
[0, 0, 0, 1]
[1, 0, 0, 0]
[0, 0, 1, 0]

Process finished with exit code 0
```