Programiz

Python Online Compiler

main.py

Share Run

```python
m = 2
n = 2
N = 2
i = 0
j = 0
directions = [(-1, 0), (1, 0), (0, -1), (0, 1)]
dp = [[[0] * n for _ in range(m)] for _ in range(N + 1)]
dp[0][i][j] = 1

out_count = 0
for step in range(N):
    for r in range(m):
        for c in range(n):
            if dp[step][r][c] > 0:
                for dr, dc in directions:
                    nr, nc = r + dr, c + dc
                    if nr < 0 or nr >= m or nc < 0 or nc >= n:
                        out_count += dp[step][r][c]
                    else:
                        dp[step + 1][nr][nc] += dp[step][r][c]

print(out_count)

```

Output

Clear

```
6

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

ogramiz
hon Online Compiler

main.py

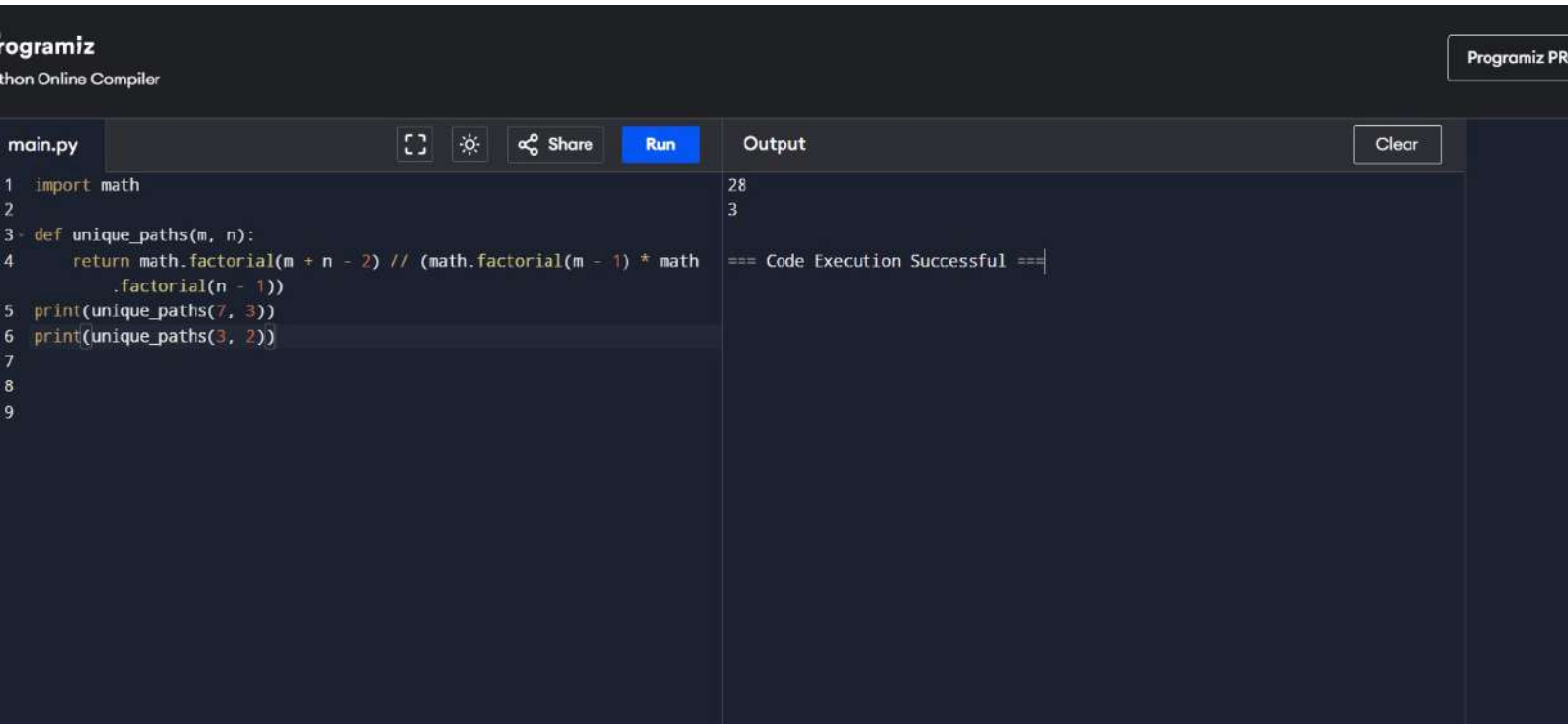
Share Run

Output Clear

```python
nums = [2, 3, 2]
def rob_linear(houses):
    prev1, prev2 = 0, 0
    for money in houses:
        prev1, prev2 = max(prev2 + money, prev1), prev1
    return prev1
if len(nums) == 1:
    print(nums[0])
else:
    result = max(rob_linear(nums[1:]), rob_linear(nums[:-1]))
    print(result)
```

```
3

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


rogramiz
thon Online Compiler
Programiz PR
main.py
Share
Run
Output
Clear
1 import math
2
3 def unique_paths(m, n):
4 return math.factorial(m + n - 2) // (math.factorial(m - 1) * math
.factorial(n - 1))
5 print(unique_paths(7, 3))
6 print(unique_paths(3, 2))
7
8
9
28
3
=== Code Execution Successful ===

Programiz

Python Online Compiler

Prog

main.py

Share

Run

Output

Clear

```python
s = "abbxxxxzzy"
n = len(s)
result = []
i = 0
while i < n:
    j = i
    while j < n and s[j] == s[i]:
        j += 1
    if j - i >= 3:
        result.append([i, j - 1])
    i = j

print(result)
```

```
[[3, 6]]

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

main.py

Share Run Output Clear

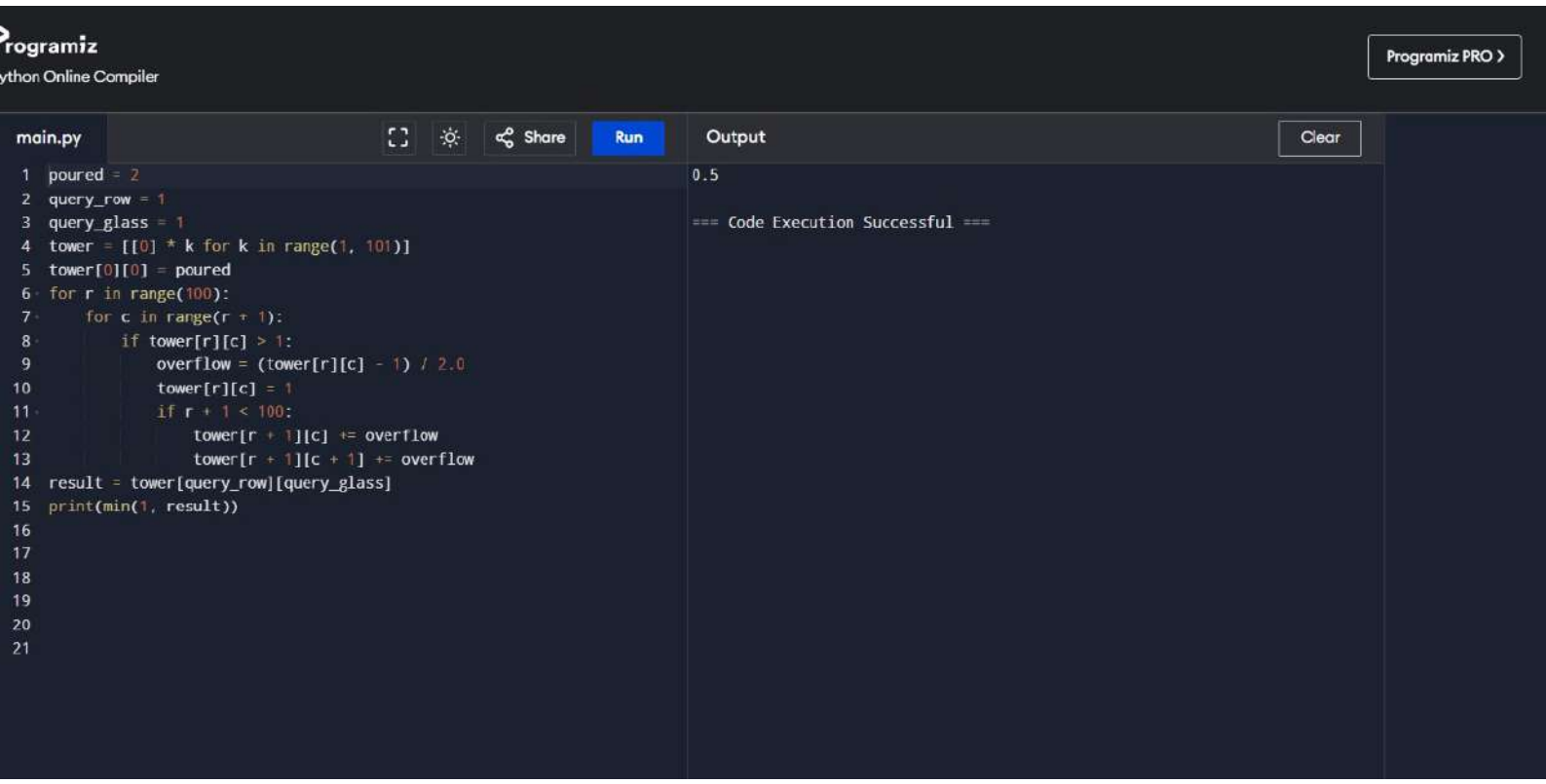
```python
board = [[0, 1, 0],
         [0, 0, 1],
         [1, 1, 1],
         [0, 0, 0]]
rows, cols = len(board), len(board[0])
next_state = [[board[r][c] for c in range(cols)] for r in range(rows)]
directions = [(-1, -1), (-1, 0), (-1, 1), (0, -1), (0, 1), (1, -1), (1, 0), (1, 1)]
for r in range(rows):
    for c in range(cols):

        live_neighbors = 0
        for dr, dc in directions:
            nr, nc = r + dr, c + dc
            if 0 <= nr < rows and 0 <= nc < cols and board[nr][nc] == 1:
                live_neighbors += 1
        if board[r][c] == 1:
            if live_neighbors < 2 or live_neighbors > 3:
                next_state[r][c] = 0
        else:
            if live_neighbors == 3:
                next_state[r][c] = 1
for row in next_state:
    print(row)

```

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

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


rogramiz
ython Online Compiler
Programiz PRO ›
main.py
Share
Run
Output
Clear
poured = 2
query_row = 1
query_glass = 1
tower = [[0] * k for k in range(1, 101)]
tower[0][0] = poured
for r in range(100):
for c in range(r + 1):
if tower[r][c] > 1:
overflow = (tower[r][c] - 1) / 2.0
tower[r][c] = 1
if r + 1 < 100:
tower[r + 1][c] += overflow
tower[r + 1][c + 1] += overflow
result = tower[query_row][query_glass]
print(min(1, result))
0.5
=== Code Execution Successful ===

main.py

Share Run

```python
def process_list(input_list):
    return sorted(input_list)

test_cases = [
    ([], []),
    ([1], [1]),
    ([7, 7, 7, 7], [7, 7, 7, 7]),
    ([-5, -1, -3, -2, -4], [-5, -4, -3, -2, -1])
]

for input_data, expected_output in test_cases:
    assert process_list(input_data) == expected_output, f"Test failed for input: {input_data}"

print("All test cases passed!")
```

Output Clear

```
All test cases passed!

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