main.py

Share Run

Output

```python
def findWays(m, n, N, i, j):
    dp = [[[0 for _ in range(N+1)] for _ in range(n)] for _ in range(m
        )]
    for step in range(1, N+1):
        for x in range(m):
            for y in range(n):
                ways = (
                    (dp[x-1][y][step-1] if x > 0 else 1) +
                    (dp[x+1][y][step-1] if x < m-1 else 1) +
                    (dp[x][y-1][step-1] if y > 0 else 1) +
                    (dp[x][y+1][step-1] if y < n-1 else 1)
                )
                dp[x][y][step] = ways
    return dp[i][j][N]
m = 2
n = 2
N = 2
i = 0
j = 0
print(findWays(m, n, N, i, j))
```

```
6

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

main.py

Share Run

Output

```python
def rob(nums):
    def rob_line(houses):
        n = len(houses)
        if n == 0:
            return 0
        if n == 1:
            return houses[0]
        dp = [0] * n
        dp[0] = houses[0]
        dp[1] = max(houses[0], houses[1])
        for i in range(2, n):
            dp[i] = max(dp[i - 1], dp[i - 2] + houses[i])
        return dp[-1]
    n = len(nums)
    if n == 0:
        return 0
    if n == 1:
        return nums[0]
    max1 = rob_line(nums[:-1])
    max2 = rob_line(nums[1:])
    return max(max1, max2)
houses = [2, 3, 2]
print(rob(houses))
```

```
3

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

main.py

Share Run

Output

```python
def climbStairs(n):
    a,b=1,1
    for i in range(n-1):
        a,b=b,a+b
    return b
print(climbStairs(4))
print(climbStairs(3))
```

```
5
3

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

main.py

Share

Run

Output

```
1  import math
2  m,n=3,2
3  print(math.comb(m+n-2,m-1))
```

```
3

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

main.py

Share Run

```python
def largeGroupPositions(S):
    result = []
    i = 0
    n = len(S)
    while i < n:
        start = i
        while i < n - 1 and S[i] == S[i + 1]:
            i += 1
        end = i
        if end - start + 1 >= 3:
            result.append([start, end])
        i += 1
    return result
S = "abbxxxxzyy"
print(largeGroupPositions(S))

```

Output

```
[[3, 6]]

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

main.py

Share Run

```python
def gameOfLife(board):
    m, n = len(board), len(board[0])
    directions = [(-1, -1), (-1, 0), (-1, 1),
                  (0, -1),          (0, 1),
                  (1, -1), (1, 0), (1, 1)]
    def countLiveNeighbors(r, c):
        count = 0
        for dr, dc in directions:
            nr, nc = r + dr, c + dc
            if 0 <= nr < m and 0 <= nc < n and abs(board[nr][nc]) == 1:
                count += 1
        return count
    for r in range(m):
        for c in range(n):
            live_neighbors = countLiveNeighbors(r, c)
            if board[r][c] == 1 and (live_neighbors < 2 or live_neighbors > 3):
                board[r][c] = -1  # Live to dead
            if board[r][c] == 0 and live_neighbors == 3:
                board[r][c] = 2  # Dead to live
    for r in range(m):
        for c in range(n):
            if board[r][c] == -1:
                board[r][c] = 0
            if board[r][c] == 2:
                board[r][c] = 1
board = [ [0, 1, 0],[0, 0, 1],[1, 1, 1],[0, 0, 0]]
gameOfLife(board)
print(board)
```

Output

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

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

main.py

Share Run

Output

```python
def champagne_tower(poured, query_row, query_glass):
    glasses = [[0] * (r + 1) for r in range(query_row + 1)]
    glasses[0][0] = poured
    for r in range(query_row):
        for c in range(r + 1):
            if glasses[r][c] > 1:
                overflow = glasses[r][c] - 1
                glasses[r][c] = 1
                glasses[r + 1][c] += overflow / 2.0
                glasses[r + 1][c + 1] += overflow / 2.0
    return min(1, glasses[query_row][query_glass])
poured = 4
query_row = 2
query_glass = 1
result = champagne_tower(poured, query_row, query_glass)
print(f"Glass at row {query_row}, glass {query_glass} has {result:.5f}
    cups of champagne.")


```

```
Glass at row 2, glass 1 has 0.50000 cups of champagne.

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