

시뮬레이션

작은 의미로는 문제가 제시한 규칙에 따라 개체를 이동시키는 알고리즘을 말하며, 큰 의미로는 문제가 요구하는 대로 시행되도록 코드를 구현하는 알고리즘

4방향 탐색 방법

	0	1	2	3	4
0					
1					
2					
3					
4					

방향배열을 만들어 네 방향을 탐색하는 방법

`dx=[-1, 0, 1, 0]`

`dy=[0, 1, 0, -1]`


```
for i in range(5):
    for j in range(5):
        for k in range(4):
            nx = i + dx[k]
            ny = j + dy[k]
            if nx >= 0 and nx < 5 and ny >= 0 and ny < 5:
                if nums[nx][ny] <= nums[i][j]:
                    flag = False
```

웅덩이

매개변수 `nums`에 n 행 n 열의 이차원 배열에 격자판 정보가 주어집니다.

각 격자에는 그 지역의 높이가 쓰여있습니다. 각 지역은 상하좌우 인접한 지역의 숫자가 모두 자신보다 클 경우 이 지역을 웅덩이 지역이라고 합니다.

격자의 가장자리는 1000으로 초기화 되었다고 가정한다.

만약 5×5 이차원 배열의 격자판 정보다 아래와 같다면 총 웅덩이의 수는 5개입니다.

1000	1000	1000	1000	1000	1000	1000
1000	10	20	50	30	20	1000
1000	20	30	50	70	90	1000
1000	10	15	25	80	35	1000
1000	25	35	40	55	80	1000
1000	30	20	35	40	90	1000
1000	1000	1000	1000	1000	1000	1000

주어진 격자에 웅덩이가 몇 개 있는지 찾아 그 개수를 반환하는 프로그램을 작성하세요.

입출력 예:


nums	answer
[[10, 20, 50, 30, 20], [20, 30, 50, 70, 90], [10, 15, 25, 80, 35], [25, 35, 40, 55, 80], [30, 20, 35, 40, 90]]	5
[[80, 25, 10, 65, 100], [20, 10, 32, 70, 33], [45, 10, 88, 9, 90], [10, 35, 10, 55, 66], [10, 84, 65, 88, 99]]	4
[[33, 22, 55, 65, 55], [55, 88, 99, 12, 19], [18, 33, 25, 57, 77], [46, 78, 54, 55, 99], [33, 25, 47, 85, 17]]	6

제한사항:

- $3 \leq n \leq 10$
- 배열 `nums`의 원소는 자연수입니다. $1 \leq \text{nums}[i][j] \leq 100$

청소 로봇(ver 1)

이차원 배열 격자판 0행 0열이 청소 로봇의 시작위치입니다.

	0	1	2	3	4	
0						
1						
2					
3						
4						
						⋮

청소 로봇은 다음 규칙에 따라 이동합니다.

1. 'U' 명령은 로봇이 위쪽으로 한 칸 이동합니다.
2. 'R' 명령은 로봇이 오른쪽으로 한 칸 이동합니다.
3. 'L' 명령은 로봇이 왼쪽으로 한 칸 이동합니다.
4. 'D' 명령은 로봇이 아래쪽으로 한 칸 이동합니다.

매개변수 moves에 청소 로봇에 명령을 내린 문자들이 차례대로 나열된 명령 문자열이 주어진다면 이 명령 문자열을 청소 로봇이 모두 수행했을 때 최종 위치를 반환하는 프로그램을 작성하세요. 격자판 밖으로 벗어나는 명령은 주어지지 않습니다.

입출력 예:

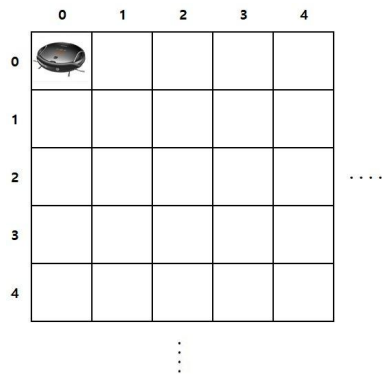
moves	answer
'RRRDDLU'	[2, 2]
'DDRRRDDLL'	[5, 1]
'RRRRRRDDDDDDUULLL'	[4, 3]
'RRRRDDDRRDDLLUU'	[3, 4]

제한사항:

- moves의 길이는 100을 넘지 않습니다.
- 2차원 배열 격자판의 크기는 100*100입니다.

청소 로봇(ver 2)

$n \times n$ 크기의 이차원 배열 격자판 0행 0열이 청소 로봇의 시작위치입니다.



청소 로봇은 다음 규칙에 따라 이동합니다.

1. 'U' 명령은 로봇이 위쪽으로 한 칸 이동합니다.
2. 'R' 명령은 로봇이 오른쪽으로 한 칸 이동합니다.
3. 'L' 명령은 로봇이 왼쪽으로 한 칸 이동합니다.
4. 'D' 명령은 로봇이 아래쪽으로 한 칸 이동합니다.
5. 만약 로봇이 명령을 수행할 경우 격자판 밖으로 나가는 경우라면 로봇은 해당 명령을 수행하지 않고 무시합니다.

매개변수 n 에 격자판 크기가 주어지고, moves에 청소 로봇에 명령을 내린 문자들이 차례대로 나열된 명령 문자열이 주어지면 청소 로봇이 최종적으로 멈춘 위치를 반환하는 프로그램을 작성하세요.

입출력 예:

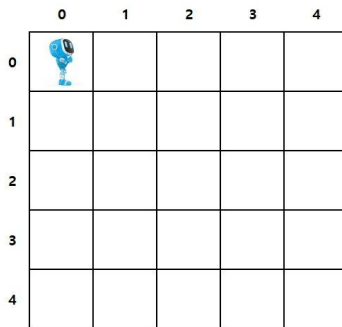
n	moves	answer
5	'RRRDDDUUUUUUL'	[0, 2]
7	'DDDRRRD DLL'	[5, 1]
5	'RRRRRDDDDDU'	[3, 4]
6	'RRRRDDDRRDDLLUU'	[3, 3]

제한사항:

- moves의 길이는 100을 넘지 않습니다.
- $3 \leq n \leq 50$

로봇의 이동

이차원 배열 격자판 0행 0열에 로봇이 3시 방향을 보고 있습니다.



로봇은 다음 규칙에 따라 이동합니다.

1. 'G' 명령을 주면 보고 있는 방향으로 한 칸 이동합니다. 격자 밖으로 나가는 명령은 하지 않습니다.
2. 'R' 명령을 주면 오른쪽으로 90도 회전합니다.
3. 'L' 명령을 주면 왼쪽으로 90도 회전합니다.

매개변수 moves에 로봇에 명령을 내린 문자들이 차례대로 나열된 명령 문자열이 주어지면 이 명령 문자열을 로봇이 모두 수행했을 때 최종 위치를 반환하는 프로그램을 작성하세요.

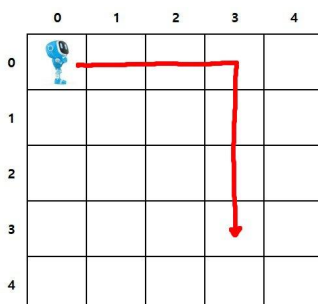
입출력 예:

moves	answer
'GGGRGGG'	[3, 3]
'GGRGGG'	[3, 2]
'GGGGGGGRGGGRGGRLGGG'	[0, 2]
'GGLLLGLGGG'	[1, 5]

제한사항:

- moves의 길이는 100을 넘지 않습니다.
- 2차원 배열 격자판의 크기는 100*100입니다.

입력예제 1 설명 :



프로그래머스 Lv.0 - 안전지대

<https://school.programmers.co.kr/learn/courses/30/lessons/120866>

정답코드

```
def solution(board):
    n = len(board)
    answer = n*n
    dx = [-1, -1, 0, 1, 1, 1, 0, -1]
    dy = [0, 1, 1, 1, 0, -1, -1, -1]
    cnt = 0
    for i in range(n):
        for j in range(n):
            if board[i][j] == 1:
                cnt += 1
                for k in range(8):
                    x = i + dx[k]
                    y = j + dy[k]
                    if x >= 0 and x < n and y >= 0 and y < n and board[x][y] == 0:
                        cnt += 1
                        board[x][y] = 2
    return answer-cnt
```

프로그래머스 Lv.1 - 공원 산책

<https://school.programmers.co.kr/learn/courses/30/lessons/172928>

정답코드

```
def solution(park, routes):
    dx = [-1, 0, 1, 0]
    dy = [0, 1, 0, -1]
    d = {'N': 0, 'E' : 1, 'S' : 2, 'W' : 3}
    n = len(park)
    m = len(park[0])
    x, y = 0, 0
    for i in range(n):
        for j in range(m):
            if park[i][j] == 'S':
                x, y = i, j
    for tmp in routes:
        cmd, dis = tmp.split(' ')
        flag = 0
        for i in range(1, int(dis)+1):
            nx, ny = x+dx[d[cmd]]*i, y+dy[d[cmd]]*i
            if nx < 0 or nx >= n or ny < 0 or ny >= m or park[nx][ny] == 'X':
                flag = 1
                break
        if flag == 0:
            x, y = nx, ny
    return [x, y]
```

프로그래머스 Lv.1 - 키패드 누르기

<https://school.programmers.co.kr/learn/courses/30/lessons/67256>

정답코드

```
def solution(numbers, hand):
    left = 10
    right = 11
    answer = ""
    dist = [[0, 0, 3, 0, 0, 2, 0, 0, 1, 0],
            [4, 0, 1, 0, 0, 2, 0, 0, 3, 0],
            [3, 0, 0, 0, 0, 1, 0, 0, 2, 0],
            [4, 0, 1, 0, 0, 2, 0, 0, 3, 0],
            [3, 0, 2, 0, 0, 1, 0, 0, 2, 0],
            [2, 0, 1, 0, 0, 0, 0, 0, 1, 0],
            [3, 0, 2, 0, 0, 1, 0, 0, 2, 0],
            [2, 0, 3, 0, 0, 2, 0, 0, 1, 0],
            [1, 0, 2, 0, 0, 1, 0, 0, 0, 0],
            [2, 0, 3, 0, 0, 2, 0, 0, 1, 0],
            [1, 0, 4, 0, 0, 3, 0, 0, 2, 0],
            [1, 0, 4, 0, 0, 3, 0, 0, 2, 0]]
    for i in numbers:
        if i in [1, 4, 7]:
            left = i
            answer += "L"
        elif i in [3, 6, 9]:
            right = i
            answer += "R"
        else:
            if dist[left][i] < dist[right][i]:
                left = i
                answer += "L"
            elif dist[left][i] > dist[right][i]:
                right = i
                answer += "R"
            elif hand == "left":
                left = i
                answer += "L"
            else:
                right = i
                answer += "R"
    return answer
```