구현_Quiz

구현 알고리즘

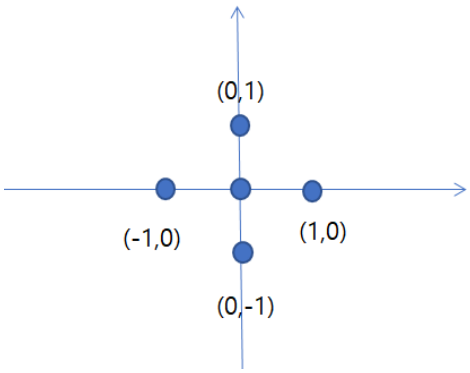
#1. 상하좌우

```
# N 입력받기
n = int(input())
x, y = 1, 1
plans = input().split()

# L, R, U, D에 따른 이동 방향
dx = [0, 0, -1, 1]
dy = [-1, 1, 0, 0]
move_types = ['L', 'R', 'U', 'D']

# 이동 계획을 하나씩 확인
for plan in plans:
    # 이동 후 좌표 구하기
    for i in range(len(move_types)):
        if plan == move_types[i]:
            nx = x + dx[i]
            ny = y + dy[i]
    # 공간을 벗어나는 경우 무시
    if nx < 1 or ny < 1 or nx > n or ny > n:
        continue
    # 이동 수행
    x, y = nx, ny

print(x, y)
```



#2. 게임 개발

```
# N, M을 공백을 기준으로 구분하여 입력받기
n, m = map(int, input().split())
```

```
# 방문한 위치를 저장하기 위한 맵을 생성하여 0으로 초기화
d = [[0] * m for _ in range(n)]   //리스트컴프리헨션으로 초기화
# 현재 캐릭터의 X 좌표, Y 좌표, 방향을 입력받기
x, y, direction = map(int, input().split())
d[x][y] = 1 # 현재 좌표 방문 처리 //가본칸&바다는 1로 표기한다고 했기 때문

# 전체 맵 정보를 입력받기
array = []
for i in range(n):
    array.append(list(map(int, input().split())))

# 북, 동, 남, 서 방향 정의
dx = [-1, 0, 1, 0]
dy = [0, 1, 0, -1]
  //0,1,2,3

# 왼쪽으로 회전
def turn_left():
    global direction  //반시계방향으로 회전한다고 했기 때문에
    direction -= 1
    if direction == -1:
        direction = 3

# 시뮬레이션 시작
count = 1
turn_time = 0
while True:
    # 왼쪽으로 회전
    turn_left()
    nx = x + dx[direction]
    ny = y + dy[direction]
    # 회전한 이후 정면에 가보지 않은 칸이 존재하는 경우 이동
    if d[nx][ny] == 0 and array[nx][ny] == 0:
        d[nx][ny] = 1  //현재좌표는 방문 처리
        x = nx
        y = ny
        count += 1   //방문한 칸의 수는 1
        turn_time = 0   //회전한 횟수(이후 turn_time=4일때를 위한 변수)
        continue
    # 회전한 이후 정면에 가보지 않은 칸이 없거나 바다인 경우
    else:
        turn_time += 1
    # 네 방향 모두 갈 수 없는 경우
    if turn_time == 4:    //4번 모두 돌았다면
        nx = x - dx[direction]
        ny = y - dy[direction]
        # 뒤로 갈 수 있다면 이동하기
        if array[nx][ny] == 0:
            x = nx
            y = ny
        # 뒤가 바다로 막혀있는 경우
        else:
            break
        turn_time = 0

# 정답 출력
print(count)
```