```
# Define a linked list node for each move
class MoveNode:
    def __init__(self, move):
        self.move = move
        self.next = None

# Print the path by traversing the linked list
def printPath(head):
    current = head
    while current:
        print(current.move)
        current = current.next

# Main logic to find shortest path using linked list
def savePrincess(grid_size, grid):
    bot_x = bot_y = 0
    princess_x = princess_y = 0

    # Locate bot and princess positions in the grid
    for i in range(grid_size):
        for j in range(grid_size):
            if grid[i][j] == 'm':
                bot_x, bot_y = i, j
            elif grid[i][j] == 'p':
                princess_x, princess_y = i, j

    head = tail = None  # Start of linked list
```

```
    # Vertical movement
    for _ in range(abs(bot_x - princess_x)):
        move = "UP" if bot_x > princess_x else "DOWN"
        node = MoveNode(move)
        if head is None:
            head = tail = node
        else:
            tail.next = node
            tail = node

    # Horizontal movement
    for _ in range(abs(bot_y - princess_y)):
        move = "LEFT" if bot_y > princess_y else "RIGHT"
        node = MoveNode(move)
        if head is None:
            head = tail = node
        else:
            tail.next = node
            tail = node

    # Output the path
    printPath(head)

# ------------------------
# ▼ Input Handling Section
# ------------------------

N = int(input("Enter grid size (odd number between 3 and 99): "))
```

```
grid = []


print("Enter grid rows:")
for _ in range(N):
    row = input().strip()
    grid.append(row)


savePrincess(N, grid)
```