```
class ChessGame:
    def __init__(self):
        self.board = [[None for _ in range(8)] for _ in range(8)]
        self.moves = []
    def is_valid_move(self, start, end):
        sx, sy = start
        ex, ey = end
        return 0 <= ex < 8 and 0 <= ey < 8 and (sx != ex or sy != ey)
    def make_move(self, start, end):
        if self.is_valid_move(start, end):
            self.moves.append((start, end))
            self.board[end[0]][end[1]] = self.board[start[0]][start[1]]
            self.board[start[0]][start[1]] = None
            return True
        return False
    def undo_move(self):
        if self.moves:
            start, end = self.moves.pop()
            self.board[start[0]][start[1]] = self.board[end[0]][end[1]]
            self.board[end[0]][end[1]] = None
    def solve(self, depth=0):
        if depth == 3:
            return True
        for i in range(8):
            for j in range(8):
                if self.board[i][j]:
                    for dx in [-1, 1]:
                        for dy in [-1, 1]:
```

```
                    new_x, new_y = i + dx, j + dy
                    if self.make_move((i, j), (new_x, new_y)):
                        if self.solve(depth + 1):
                            return True
                        self.undo_move()
        return False
game = ChessGame()
x, y = map(int, input("Enter your piece position (row col): ").split())
game.board[x][y] = input("Enter piece type (e.g., R for Rook, N for Knight): ")
if game.solve():
    print("Solution Found!")
else:
    print("No Solution Possible.")



class ChessGame:
    def __init__(self):
        self.board = [[None for _ in range(8)] for _ in range(8)]
        self.moves = []

    def is_valid_move(self, start, end):
        sx, sy = start
        ex, ey = end
        return 0 <= ex < 8 and 0 <= ey < 8 and (sx != ex or sy != ey)

    def make_move(self, start, end):
        if self.is_valid_move(start, end):
            self.moves.append((start, end))
```

```
            self.board[end[0]][end[1]] = self.board[start[0]][start[1]]
            self.board[start[0]][start[1]] = None
            return True
        return False

    def undo_move(self):
        if self.moves:
            start, end = self.moves.pop()
            self.board[start[0]][start[1]] = self.board[end[0]][end[1]]
            self.board[end[0]][end[1]] = None

    def solve(self, depth=0):
        if depth == 3:
            return True

        for i in range(8):
            for j in range(8):
                if self.board[i][j]:
                    for dx in [-1, 1]:
                        for dy in [-1, 1]:
                            new_x, new_y = i + dx, j + dy
                            if self.make_move((i, j), (new_x, new_y)):
                                if self.solve(depth + 1):
                                    return True
                                self.undo_move()
        return False

game = ChessGame()
```

```
x, y = map(int, input("Enter your piece position (row col): ").split())
game.board[x][y] = input("Enter piece type (e.g., R for Rook, N for Knight): ")
if game.solve():
    print("Solution Found!")
else:
    print("No Solution Possible.")


class ChessGame:
    def __init__(self):
        self.board = [[None for _ in range(8)] for _ in range(8)]
        self.moves = []

    def is_valid_move(self, start, end):
        sx, sy = start
        ex, ey = end
        return 0 <= ex < 8 and 0 <= ey < 8 and (sx != ex or sy != ey)

    def make_move(self, start, end):
        if self.is_valid_move(start, end):
            self.moves.append((start, end))
            self.board[end[0]][end[1]] = self.board[start[0]][start[1]]
            self.board[start[0]][start[1]] = None
            return True
        return False

    def undo_move(self):
        if self.moves:
            start, end = self.moves.pop()
```

```
        self.board[start[0]][start[1]] = self.board[end[0]][end[1]]
        self.board[end[0]][end[1]] = None

    def solve(self, depth=0):
        if depth == 3:
            return True

        for i in range(8):
            for j in range(8):
                if self.board[i][j]:
                    for dx in [-1, 1]:
                        for dy in [-1, 1]:
                            new_x, new_y = i + dx, j + dy
                            if self.make_move((i, j), (new_x, new_y)):
                                if self.solve(depth + 1):
                                    return True
                                self.undo_move()
        return False

game = ChessGame()
x, y = map(int, input("Enter your piece position (row col): ").split())
game.board[x][y] = input("Enter piece type (e.g., R for Rook, N for Knight): ")
if game.solve():
    print("Solution Found!")
else:
    print("No Solution Possible.")
```

```
class ChessGame:
    def __init__(self):
        self.board = [[None for _ in range(8)] for _ in range(8)]
        self.moves = []

    def is_valid_move(self, start, end):
        sx, sy = start
        ex, ey = end
        return 0 <= ex < 8 and 0 <= ey < 8 and (sx != ex or sy != ey)

    def make_move(self, start, end):
        if self.is_valid_move(start, end):
            self.moves.append((start, end))
            self.board[end[0]][end[1]] = self.board[start[0]][start[1]]
            self.board[start[0]][start[1]] = None
            return True
        return False

    def undo_move(self):
        if self.moves:
            start, end = self.moves.pop()
            self.board[start[0]][start[1]] = self.board[end[0]][end[1]]
            self.board[end[0]][end[1]] = None

    def solve(self, depth=0):
        if depth == 3:
            return True
```

```
        for i in range(8):
            for j in range(8):
                if self.board[i][j]:
                    for dx in [-1, 1]:
                        for dy in [-1, 1]:
                            new_x, new_y = i + dx, j + dy
                            if self.make_move((i, j), (new_x, new_y)):
                                if self.solve(depth + 1):
                                    return True
                                self.undo_move()
        return False

game = ChessGame()
x, y = map(int, input("Enter your piece position (row col): ").split())
game.board[x][y] = input("Enter piece type (e.g., R for Rook, N for Knight): ")
if game.solve():
    print("Solution Found!")
else:
    print("No Solution Possible.")
```