FINALS KING PYTHON

```
def outOfBounds(y, x):
 return y < 0 or x < 0 or y > 7 or x > 7
def getPos(string):
 return (8-int(string[1]), ord(string[0]) - ord('a'))
def find_king_status(pieces):
 pieces = pieces.strip().split(' ')

kingPos = (0, 0)
kingPos (0, 0)
 board = [[False for i in range(8)] for j in range(8)]

 whitePos = set()
 for piece in pieces:
 pos = getPos(piece[1:])
 if piece[0] != 'K':
 whitePos.add(pos)

 for piece in pieces:
 pos = getPos(piece[1:])
 if piece[0] == 'Q' or piece[0] == 'R':
 curPos = (pos[0]-1, pos[1])
 while not (outOfBounds(*curPos) or curPos in whitePos):
 board[curPos[0]][curPos[1]] = True
 curPos = (curPos[0]-1, curPos[1])
 if curPos in whitePos:
 board[curPos[0]][curPos[1]] = True
 curPos = (pos[0], pos[1]+1)
```

```
while not (outOfBounds(*curPos) or curPos in whitePos):
board[curPos[0]][curPos[1]] = True
curPos = (curPos[0], curPos[1]+1)
if curPos in whitePos:
board[curPos[0]][curPos[1]] = True
curPos = (pos[0]+1, pos[1])
while not (outOfBounds(*curPos) or curPos in whitePos):
board[curPos[0]][curPos[1]] = True
curPos = (curPos[0]+1, curPos[1])
if curPos in whitePos:
board[curPos[0]][curPos[1]] = True
curPos = (pos[0], pos[1]-1)
while not (outOfBounds(*curPos) or curPos in whitePos):
board[curPos[0]][curPos[1]] = True
curPos = (curPos[0], curPos[1]-1)
if curPos in whitePos:
board[curPos[0]][curPos[1]] = True
if piece[0] == 'Q' or piece[0] == 'B':
curPos = (pos[0]-1, pos[1]-1)
while not (outOfBounds(*curPos) or curPos in whitePos):
board[curPos[0]][curPos[1]] = True
curPos = (curPos[0]-1, curPos[1]-1)
if curPos in whitePos:
board[curPos[0]][curPos[1]] = True
curPos = (pos[0]-1, pos[1]+1)
while not (outOfBounds(*curPos) or curPos in whitePos):
board[curPos[0]][curPos[1]] = True
curPos = (curPos[0]-1, curPos[1]+1)
if curPos in whitePos:
```

```
 board[curPos[0]][curPos[1]] = True
 curPos = (pos[0]+1, pos[1]-1)
 while not (outOfBounds(*curPos) or curPos in whitePos):
 board[curPos[0]][curPos[1]] = True
 curPos = (curPos[0]+1, curPos[1]-1)
 if curPos in whitePos:
 board[curPos[0]][curPos[1]] = True
 curPos = (pos[0]+1, pos[1]+1)
 while not (outOfBounds(*curPos) or curPos in whitePos):
 board[curPos[0]][curPos[1]] = True
 curPos = (curPos[0]+1, curPos[1]+1)
 if curPos in whitePos:
 board[curPos[0]][curPos[1]] = True
 if piece[0] == 'P':
 curPos = (pos[0]-1, pos[1]-1)
 if not outOfBounds(*curPos):
board[curPos[0]][curPos[1]] = True
board[curPos[0]][curPos[1]] = True
 curPos = (pos[0]-1, pos[1]+1)
 if not outOfBounds(*curPos):
 board[curPos[0]][curPos[1]] = True
 if piece[0] == 'N':
 options = [1, -1, 2, -2]
 for i in options:
 for j in options:
 if abs(i) + abs(j) == 3:
 curPos = (pos[0]+i, pos[1]+j)
 if not outOfBounds(*curPos):
 board[curPos[0]][curPos[1]] = True
```

```
if piece[0] == 'K':
kingPos = pos

for i in board:
print([1 if i[j] else 0 for j in range(len(i))])

check = [board[kingPos[0]][kingPos[1]], True]
options = [(0, 1), (-1, 1), (-1, 0), (-1, -1), (0, -1), (1, -1), (1, 0),
(1, 1)]
for i in options:
curPos = (kingPos[0]+i[0], kingPos[1]+i[1])
if not outOfBounds(*curPos):
check[1] &= board[curPos[0]][curPos[1]]

if not check[0] and not check[1]:
return "SAFE"
elif check[0] and not check[1]:
return "CHECK"
elif check[0] and check[1]:
return "CHECKMATE"
else:
return "STALEMATE"
```