In [1]:
```python
def initialize_board():
    return [[' ' for _ in range(3)] for _ in range(3)]

def print_board(board):
    for row in board:
        print('|'.join(row))
        print('-' * 5)
```

In [2]:
```python
def check_winner(board, player):
    for row in board:
        if all(cell == player for cell in row):
            return True
    for col in range(3):
        if all(board[row][col] == player for row in range(3)):
            return True
    if (board[0][0] == player and board[1][1] == player and board[2][2] == player)
       (board[0][2] == player and board[1][1] == player and board[2][0] == player):
        return True
    return False
```

In [3]:
```python
def get_empty_cells(board):
    empty_cells = []
    for i in range(3):
        for j in range(3):
            if board[i][j] == ' ':
                empty_cells.append((i, j))
    return empty_cells
```

In [4]:
```python
def minimax(board, depth, is_maximizing):
    scores = {'X': 1, 'O': -1, 'tie': 0}

    winner = check_winner(board, 'X') or check_winner(board, 'O')
    if winner:
        return scores['X'] if winner == 'X' else scores['O']
    if not get_empty_cells(board):
        return scores['tie']

    if is_maximizing:
        best_score = -float('inf')
        for i, j in get_empty_cells(board):
            board[i][j] = 'X'
            score = minimax(board, depth + 1, False)
            board[i][j] = ' '
            best_score = max(best_score, score)
        return best_score
    else:
        best_score = float('inf')
        for i, j in get_empty_cells(board):
            board[i][j] = 'O'
            score = minimax(board, depth + 1, True)
            board[i][j] = ' '
            best_score = min(best_score, score)
        return best_score
```

In [5]:
```
def get_computer_move(board):
    best_score = -float('inf')
    best_move = None
    for i, j in get_empty_cells(board):
        board[i][j] = 'X'
        score = minimax(board, 0, False)
        board[i][j] = ' '
        if score > best_score:
            best_score = score
            best_move = (i, j)
    return best_move

board = initialize_board()
current_player = 'O'  # Human starts
while True:
    print_board(board)
    if current_player == 'O':
        row, col = map(int, input("Enter your move (row, col): ").split(','))
        if board[row][col] != ' ':
            print("Invalid move. Try again.")
            continue
        board[row][col] = 'O'
    else:
        print("Computer's turn...")
        row, col = get_computer_move(board)
        board[row][col] = 'X'

    winner = check_winner(board, current_player)
    if winner:
        print_board(board)
        print(f"{current_player} wins!")
        break
    if not get_empty_cells(board):
        print_board(board)
        print("It's a tie!")
        break # Move break to end the while loop when it's a tie

    current_player = 'X' if current_player == 'O' else 'O' # Now correctly indentea
```

```
 | |
-----
 | |
-----
 | |
-----
```

```
O| |
-----
 | |
-----
 | |
-----
Computer's turn...
O|X|
-----
 | |
-----
 | |
-----
O|X|O
-----
 | |
-----
 | |
-----
Computer's turn...
O|X|O
-----
X| |
-----
 | |
-----
O|X|O
-----
X| |O
-----
 | |
-----
Computer's turn...
O|X|O
-----
X| |O
-----
 | |X
-----
O|X|O
-----
X|O|O
-----
 | |X
-----
Computer's turn...
O|X|O
-----
X|O|O
-----
X| |X
-----
```

```
O|X|O
-----
X|O|O
-----
X|O|X
-----
It's a tie!
```

In [ ]: