AI LAB

LAB 10

The alpha–beta search algorithm.

CODE:

```
import math

# ----- Game helper functions -----

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

def check_winner(board):
    # Check rows and columns
    for i in range(3):
        if board[i][0] == board[i][1] == board[i][2] != ' ':
            return board[i][0]
        if board[0][i] == board[1][i] == board[2][i] != ' ':
            return board[0][i]
    # Check diagonals
    if board[0][0] == board[1][1] == board[2][2] != ' ':
        return board[0][0]
    if board[0][2] == board[1][1] == board[2][0] != ' ':
        return board[0][2]
    return None

def is_moves_left(board):
    return any(' ' in row for row in board)

# ----- Minimax with Alpha-Beta Pruning -----

def alphabeta(board, depth, alpha, beta, is_maximizing):
    winner = check_winner(board)
    if winner == 'X':
        return 1
    elif winner == 'O':
        return -1
    elif not is_moves_left(board):
        return 0
```

```
    if is_maximizing:  # AI's turn (X)
        max_eval = -math.inf
        for i in range(3):
            for j in range(3):
                if board[i][j] == ' ':
                    board[i][j] = 'X'
                    eval = alphabeta(board, depth + 1, alpha, beta, False)
                    board[i][j] = ' '
                    max_eval = max(max_eval, eval)
                    alpha = max(alpha, eval)
                    if beta <= alpha:
                        break  # 🔥 prune branch
        return max_eval

    else:  # Human's turn (O)
        min_eval = math.inf
        for i in range(3):
            for j in range(3):
                if board[i][j] == ' ':
                    board[i][j] = 'O'
                    eval = alphabeta(board, depth + 1, alpha, beta, True)
                    board[i][j] = ' '
                    min_eval = min(min_eval, eval)
                    beta = min(beta, eval)
                    if beta <= alpha:
                        break  # 🔥 prune branch
        return min_eval

def find_best_move(board):
    best_val = -math.inf
    best_move = None

    for i in range(3):
        for j in range(3):
            if board[i][j] == ' ':
                board[i][j] = 'X'
                move_val = alphabeta(board, 0, -math.inf, math.inf, False)
                board[i][j] = ' '
                if move_val > best_val:
                    best_val = move_val
                    best_move = (i, j)
    return best_move

# ----- Game Loop -----
```

```
def play_game():
    board = [[' ' for _ in range(3)] for _ in range(3)]
    print("Welcome to Tic Tac Toe (Alpha-Beta Edition)!")
    print("You are 'O' and the AI is 'X'.")
    print_board(board)

    while True:
        # Human move
        move = input("Enter your move (row and column, e.g. 1 3):
").split()
        if len(move) != 2:
            print("Invalid input! Please enter row and column numbers (1-
3).")
            continue
        i, j = int(move[0]) - 1, int(move[1]) - 1
        if not (0 <= i < 3 and 0 <= j < 3):
            print("Thats outside the board. Try again.")
            continue
        if board[i][j] != ' ':
            print("That spot is already taken. Try again.")
            continue
        board[i][j] = 'O'

        print_board(board)
        if check_winner(board) == 'O':
            print("🎉 You win!")
            break
        if not is_moves_left(board):
            print("It's a draw!")
            break

        # AI move
        print("AI is thinking...")
        ai_move = find_best_move(board)
        board[ai_move[0]][ai_move[1]] = 'X'
        print_board(board)

        if check_winner(board) == 'X':
            print("🤖 AI wins!")
            break
        if not is_moves_left(board):
            print("It's a draw!")
            break
```

```python
# Run the game
if __name__ == "__main__":
    play_game()
```

Output:

```
Welcome to Tic Tac Toe (Alpha-Beta Edition)!
You are 'O' and the AI is 'X'.
 | |
-----
 | |
-----
 | |
-----
Enter your move (row and column, e.g. 1 3): 1 1
O| |
-----
 | |
-----
 | |
-----
AI is thinking...
O| |
-----
 |X|
-----
 | |
-----
Enter your move (row and column, e.g. 1 3): 2 3
O| |
-----
 |X|O
-----
 | |
-----
AI is thinking...
O|X|
-----
 |X|O
-----
 | |
-----
Enter your move (row and column, e.g. 1 3): 3 3
O|X|
-----
 |X|O
-----
 | |O
```

```
AI is thinking...
O|X|
-----
 |X|O
-----
 | |
-----
Enter your move (row and column, e.g. 1 3): 3 3
O|X|
-----
 |X|O
-----
 | |O
-----
AI is thinking...
O|X|X
-----
 |X|O
-----
 | |O
-----
Enter your move (row and column, e.g. 1 3): 3 2
O|X|X
-----
 |X|O
-----
 |O|O
-----
AI is thinking...
O|X|X
-----
 |X|O
-----
X|O|O
-----
```

AI wins!