ARTIFICIAL INTELLIGENCE LAB

BSCYS-3rd Semester

Fall 2025

LAB REPORT # 8

Submitted To:**Sir Mubashir Iqbal**

Submitted by:**M.Umer**

Reg No**:(24-cys-024)**

Department:**Cyber Security (A)**

BS CYBER SECURITY PROGRAM

DEPARTMENT OF COMPUTER SCIENCE HITEC UNVERSITY TAXILA

TASK 1:

ScreenShot:

```python
import math

def alpha_beta(depth, node_index, is_max, values, alpha, beta):
    # Terminal condition (leaf node reached)
    if depth == 3:
        return values[node_index]

    # MAX player's turn
    if is_max:
        best_value = -math.inf

        for i in range(2):
            value = alpha_beta(
                depth + 1,
                node_index * 2 + i,
                False,
                values,
                alpha,
                beta
            )
            best_value = max(best_value, value)
            alpha = max(alpha, best_value)
```

```python
            # Alpha-Beta Pruning condition
            if beta <= alpha:
                break

        return best_value

    # MIN player's turn
    else:
        best_value = math.inf

        for i in range(2):
            value = alpha_beta(
                depth + 1,
                node_index * 2 + i,
                True,
                values,
                alpha,
                beta
            )
            best_value = min(best_value, value)
            beta = min(beta, best_value)
```

```python
            # Alpha-Beta Pruning condition
            if beta <= alpha:
                break

        return best_value


# Terminal node values (left to right)
terminal_values = [4, 2, -3, -6, 7, 0, 5, 8]

# Applying Alpha-Beta Pruning from root node A
optimal_value = alpha_beta(
    depth=0,
    node_index=0,
    is_max=True,
    values=terminal_values,
    alpha=-math.inf,
    beta=math.inf
)

print("Optimal value at root node A is:", optimal_value)
```

Output:

```
PS C:\Users\FAIZAN COMPUTER> & "C:/Users/FAIZAN COMPUTER/AppData/Local/Programs/Python/Python314/python.exe" "c:/Users/FAIZAN COMPUTER/alpha_beeta_pruning.py"
Optimal value at root node A is: 7
PS C:\Users\FAIZAN COMPUTER>
```

Task 2:

ScreenShot:

alpha_beeta_pruning.py tic_tac_toe_minimax.py

C: > Users > FAIZAN COMPUTER > tic_tac_toe_minimax.py > ...

```python

PLAYER_X = 'X'  # Maximizing player
PLAYER_O = 'O'  # Minimizing player
EMPTY = ' '

board = [
    ['O', 'X', 'O'],
    ['O', 'X', 'X'],
    ['X', 'O', 'X']
]

def check_winner(board):
    # Rows, columns and diagonals
    lines = board + list(zip(*board)) + [
        [board[i][i] for i in range(3)],
        [board[i][2 - i] for i in range(3)]
    ]

    for line in lines:
        if line.count(PLAYER_X) == 3:
            return 1
        if line.count(PLAYER_O) == 3:
            return -1
```

```python
    return 0

def is_full(board):
    return all(cell != EMPTY for row in board for cell in row)

def minimax(board, is_max):
    score = check_winner(board)

    if score != 0:
        return score
    if is_full(board):
        return 0  # Draw

    if is_max:
        best = -1000
        for i in range(3):
            for j in range(3):
                if board[i][j] == EMPTY:
                    board[i][j] = PLAYER_X
                    best = max(best, minimax(board, False))
                    board[i][j] = EMPTY
        return best
    else:
```

```python
    else:
        best = 1000
        for i in range(3):
            for j in range(3):
                if board[i][j] == EMPTY:
                    board[i][j] = PLAYER_O
                    best = min(best, minimax(board, True))
                    board[i][j] = EMPTY
        return best


result = minimax(board, True)

if result == 1:
    print("X will win")
elif result == -1:
    print("O will win")
else:
    print("Game will be a Draw")
```

Output:

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS                Python

PS C:\Users\FAIZAN COMPUTER> & "C:/Users/FAIZAN COMPUTER/AppData/Local/Programs/Python/Python314/python.exe" "c:
/Users/FAIZAN COMPUTER/tic_tac_toe_minimax.py"
Game will be a Draw
PS C:\Users\FAIZAN COMPUTER>
```

Conclusion:

In this lab, I learn how to applied Alpha-Beta Pruning to a minimax tree and determined the optimal value at the root node A as 7, with pruning reducing unnecessary evaluations and improving efficiency. I also applied the Minimax algorithm to analyze a nearly full Tic-Tac-Toe board, where no winning moves were available, resulting in a draw with a Minimax value of 0. Overall, the lab explain how decision-making algorithms like Minimax and Alpha-Beta Pruning can be used to evaluate game trees and strategic scenarios efficiently, highlighting both optimal decision selection and computational optimization