Date ______
Page ______

Lab-10 Minmax and Alpha-Beta Pruning

→ **Minimax for Tic-Tac-Toe algorithm:**

```
def minimax(board):
    if terminal(board):
        return None

    def max_val(board):
        if terminal(board):
            return ~~None, None~~ utility(board), None
        val = -INT_MAX
        action = None
        for move in possible_moves(board):
            v, act = min_val(result(board, move))
            if v > val:
                val = v
                action = move
                if v == 1:
                    return v, action
        return (v, action)

    def min_val(board):
        if terminal(board):
            return ~~None~~ utility(board), None
        val = INT_MAX
        action = None
        for move in possible_moves(board):
            v*, act = ~~max~~ max_val(result(board, move))
            if v < val:
                value = v
            action = move
            if value == -1:
                  return (v, action)
    return (val, action)

    if current == PLAYER:
            return max_val [board) [1]
    else:
            return min_val [board) [1]
```

Utility Functions:

```
    def terminal (board):
          if ~~not~~ (winner)(board) ~~or~~ or
                empty_cells (board) == 0:
                return True
          else:
                return False

    def result (board, action) → Returns new board with
                                   piece placed in position
                                   defined by {action}

    def utility (board):
           if winner (board) == 'X':
                  return 1
           elif winner (board) == "O":
                  return -1
           else:
                  return 0
```

→ Alpha-Beta Pruning for 8n-queens problem

```
def alabprune (board, ~~alpha~~ a = -INT_MAX, b = INT_MAX):

    ~~if not isSafe (board):~~
        ~~return False~~
    if row == N:
        return board

    ~~for col ~~in~~ ← 0 ~~to~~ to N-1~~
        ~~if Safe (board, row, col):~~
            ~~board [row] [col] = 1~~
            ~~result = ~~al~~abprune (board, row+1, a, b)~~
            ~~if res:~~
                ~~return res~~
        ~~board [row] [col] = 0~~
        ~~alpha = max (alpha, row)~~
        ~~if alpha >= beta:~~
            ~~break~~
    return ~~Return Null~~ NULL
    for col i ← 0 to N-1
        If Safe (board, row, col):
            board [row] [col] = 1
    ~~result~~ result = abprune (board, result+1,
                    a, b)
            if res:
                return res
    ~~boo~~ board [row] [col] = 0
    ~~alpha~~ a = max (~~alpha~~ a, ~~a~~ row)
        if ~~alpha~~ a >= b:
            break
```

Date
Page

return NULL

Using the algorithm to solve we can get a possible output for 8 queens as (0, 4, 7, 5, 2, 6, 1, 3)

[illegible]
3/12/24