**Project Title:**

**Checkers: A Multi-Jump Strategy Game**

**Submitted By:**

Muhammad Sohaib Baig

**Course:**

AI

**Instructor:**

**Theory:** Sir Saif

**Lab:** Sir Talha

**Submission Date:**

20-April 2025

**1. Project Overview**

**Project Topic:**

This project is based on the classic game of **Checkers**, but with a twist: we introduce a variant called **Multi Checkers**. In this version, each player starts with **10 pieces**, and a new feature called **"Multi Jump"** allows for strategic long-distance moves under specific conditions. The board is also expanded to **10x10**, and players can chain multi jumps with increased complexity.

**Objective:**

The goal of the project is to develop a strategic AI capable of playing this modified version of Checkers using **Minimax with Alpha-Beta Pruning**, and to design effective heuristics to evaluate game states. This will challenge both board control and foresight, testing the AI's ability to manage complexity in a multi-path environment.

**2. Game Description**

**Original Game Background:**

Checkers is a two-player board game played on an 8×8 board using 12 pieces per player. Players move diagonally and capture by jumping over opponent pieces. When a piece reaches the opponent’s back row, it becomes a king and can move backward.

**Innovations Introduced:**

* **Multi Jump**: Once per game, each piece can make a **multi jump** — moving over two empty squares instead of one, giving the possibility of longer, multi-jump combos.
* **Expanded Board**: The board is now **10x10**, allowing more strategic movement.
* **Warp Zones**: Special board areas (center squares) that grant bonus moves or prevent captures.
* **Impact on Complexity and Strategy:**  
  These innovations increase the branching factor of each move and require deeper foresight. The Multi Jump and Warp Zones add conditional logic, making the game more dynamic and suitable for AI exploration and evaluation.

**3. AI Approach and Methodology**

**AI Techniques to be Used:**

* **Minimax Algorithm**: Core decision-making algorithm, modified to evaluate deeper strategies.
* **Alpha-Beta Pruning**: To reduce the number of evaluated nodes and improve efficiency.

**Heuristic Design:**

* Piece value (normal vs king)
* Mobility (number of valid moves)
* Board control (center vs edge dominance)
* Proximity to promotion
* Risk-reward analysis for multi jumps and warp zone positioning

**Complexity Analysis:**

* **State Space**: Increased due to the 10x10 board and quantum mechanics
* **Time Complexity**: O(b^d) where b is the branching factor and d is depth; significantly larger than standard checkers
* Challenges include move validation for multi jumps and handling special rules dynamically.

**4. Game Rules and Mechanics**

**Modified Rules:**

* Each player has 10 pieces.
* Board size: 10x10.
* Each piece may use a **Multi Jump** move once per game.
* **Warp Zones** located at the 4 central squares offer immunity from capture or bonus movement.

**Winning Conditions:**

* A player wins by capturing all opponent pieces or blocking all legal moves.
* In case both players are left with one king and no captures after 20 moves, the game ends in a draw.

**Turn Sequence:**

* Turns alternate between players.
* On each turn, a player moves one piece.
* If a capture is possible, it must be made.

**5. Implementation Plan**

**Programming Language:**

Python

**Libraries and Tools:**

* **Pygame** – for GUI and game rendering
* **NumPy** – for board state representation and manipulation

**Milestones and Timeline:**

* **Week 1–2**: Game design and rule finalization
* **Week 3–4**: Develop AI strategy with Minimax and heuristics
* **Week 5–6**: Code base game mechanics and multi/warp logic
* **Week 7**: AI integration and testing with Pygame interface
* **Week 8**: Final testing, tuning, and report preparation

**6. References**

* Checkers AI with Minimax: <https://www.geeksforgeeks.org/minimax-algorithm-in-game-theory>
* AI in Board Games – Stanford CS221 Notes
* Pygame documentation: <https://www.pygame.org/docs/>