**AI PROJECT PROPOSAL**

**BCS -6J**

**GROUP MEMBERS: HUZAIFA NASEER [22K-4586], TAMEEMA REHMAN [22K-4169], TAHA TAHIR [22L-6754]**

**----------------------------------------------------------------------------------**

**PROJECT TITLE: “REVERSI – STRATEGIC AI SHOWDOWN”**

**1. Game Mechanics and Board Representation**

• **Board Layout:**  
Represent the board as an 12×12 grid (using a two-dimensional list or matrix). Each cell can hold values representing an empty slot, a black piece, or a white piece.

• **Rules Implementation:**  
– Initialize the board with the standard four pieces in the centre.  
– Implement functions to check valid moves: for any candidate move, verify that placing a piece will capture at least one of the opponent’s pieces in any of the eight directions.  
– Create a move execution function that flips the appropriate opponent pieces either in vertical or horizontal direction.

**2. Human vs. AI Game Loop**

• **Turn Management:**  
– Alternate turns between the human and the AI.  
– Validate human input (using a text-based interface or a GUI via Pygame/Tkinter).  
– Display the updated board state after each move.

• **Game End Conditions:**  
– Determine when no valid moves remain for both players or when the board is full.  
– Tally the pieces to decide the winner.

**3. AI Decision-Making Process**

• **State Evaluation Function:**  
– Develop a heuristic to score board states. For instance, give extra weight to corner positions and edges, which are strategically advantageous.  
– Consider the current difference in piece counts, mobility (number of possible moves), and board stability.

• **Move Generation:**  
– Create a function to generate all legal moves for the AI based on the current board state.

• **Search Algorithm (Minimax with Alpha-Beta Pruning):**  
– Implement the minimax algorithm to simulate move sequences, where the AI (maximizing player) and the human (minimizing player) alternate turns.  
– Integrate alpha-beta pruning to eliminate branches that don’t need evaluation, improving efficiency.  
– Define a search depth that balances decision quality and computational feasibility.

**4. Implementation Plan**

• **Coding Language and Environment:**  
– Use Python for the core logic.  
– Start with a text-based interface; later, you might consider adding a GUI using libraries like Pygame if desired.

• **Milestones:**

1. **Week 1-2:** Set up the board representation and implement Reversi game rules.
2. **Week 3-4:** Develop move generation and state validation functions.
3. **Week 5-6:** Build and test the AI’s decision-making process (minimax, evaluation function, alpha-beta pruning).
4. **Week 7:** Integrate human input with AI moves and refine the game loop.
5. **Week 8:** Conduct final testing, optimize performance, and prepare documentation.

**5. Additional Considerations**

• **Heuristic Tweaks:**  
– Experiment with different weights in the evaluation function. For example, emphasize corner occupancy, edge stability, or mobility, depending on the stage of the game.

• **Difficulty Levels:**  
– Consider adding adjustable difficulty by varying the search depth or heuristic aggressiveness.

• **User Experience:**  
– Provide clear instructions, an option to restart, and possibly hints to guide human players.