Artificial Intelligence – Spring 2025 Assignment 4

Human vs AI Chess Game Using Minimax Algorithm

Deadline: 20th April,2025

General Instructions:

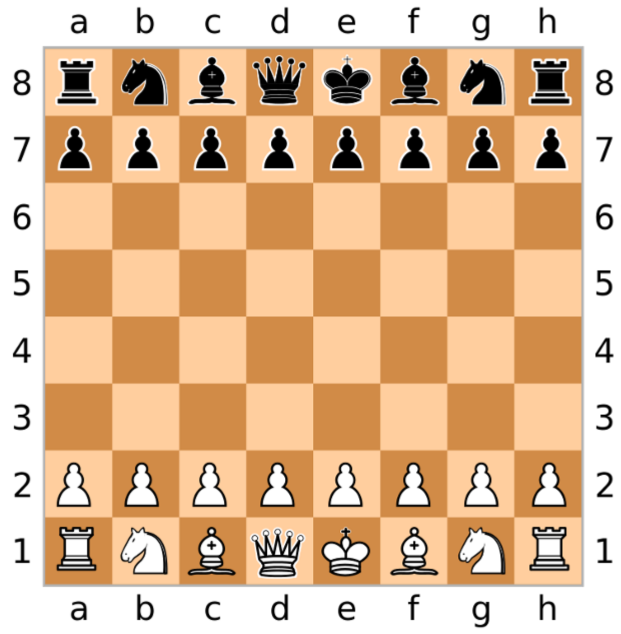
- This assignment follows the following CLOS:
    - **[CLO - 4]** General understanding of major concepts and approaches in knowledge representation, planning, learning, robotics and other AI areas.
    - **[CLO - 5]** Developing programming skills for AI applications.
- This is an individual assignment.
- The deadline will not be extended under any circumstances.
- **A pdf containing all the rules of chess game has been provided along with this assignment statement.**
- **All the components of the algorithm being used (Minimax) should be mentioned in the PDF report along with 1 test case of each game playing features mentioned in heading 3 of the assignment statement showing that you have implemented that feature in your solution.**
- Evaluation will be done on the basis of a demo. Anyone failing to appear for the demo or unable to give up to the mark demonstration will be awarded zero.

Objective:

Your task is to implement a simplified Chess game in Python with an AI opponent powered by the **Minimax algorithm with Alpha-Beta Pruning**. The game must support **Human vs AI** mode and include a **Graphical User Interface (GUI)** for interaction. All core chess mechanics must be implemented correctly.

Introduction

A game of chess has several kinds of pieces: pawns, knights, bishops, rooks, queens, and kings. These pieces are arranged on a chessboard as shown in the figure below.



A cell on the board is specified by a (row, column) tuple: rows increasing from bottom to top and columns increasing from left to right. Traditionally the black pieces are arranged in the top two rows as shown. Each chess piece can move in a specific way. In addition to moving, each chess piece can also kill a chess piece of the opposite color if it moves to its place.

There are 6 main pieces in chess:

Queen: worth 9 points.

Rook: worth 5 points.

Bishop: worth 3 points.

Knight: worth 3 points.

Pawn: worth 1 point.

King: worth infinity points, technically.

The basic rules for each chess piece are discussed in other shared documents.

The main goal of chess is to capture the king of the opponent.

Assignment Requirements:

1. Code Structure

Your solution **must follow an object-oriented approach**. The following class structure is **required** (you may create additional helper classes as needed):

- **ChessGame:** Controls the overall game loop, manages players' turns, and checks for endgame conditions.
- **Board:** Represents the 8x8 chessboard and tracks the state of all pieces.
- **Move:** Represents a move, including the source and destination squares, the moving piece, and move type (normal, capture, castling, promotion, en passant).
- **Piece:** Abstract base class for all pieces, with common methods such as get_valid_moves().
- One **separate class for each chess piece** inheriting from Piece:
  - **King, Queen, Rook, Bishop, Knight, Pawn**
- **Player**: Abstract class/interface for both player types.
- **HumanPlayer:** Handles move input through GUI interaction.
- **AIPlayer:** Implements Minimax with Alpha-Beta Pruning and uses an evaluation function.
- **Evaluation:** Contains the board evaluation heuristics used by the AI.

2. AI Implementation

- The **AIPlayer** class must implement the **Minimax algorithm**. Implementing Alpha-Beta Pruning is optional.
- The AI should search the game tree up to a fixed **depth (e.g., 2–3 plies or levels)**.
- The evaluation function must assess board positions based on:

  1. Material count (assign weights to different pieces)
  2. King safety (e.g., avoid exposing the king)
  3. Positional advantages (e.g., central control, piece mobility)

3. Gameplay Features

Your implementation must correctly handle the following gameplay mechanics:

- Turn-based play between a human and the AI.
- Legal move generation for each piece based on standard chess rules.
- Special moves:
  - **Castling** (both kingside and queenside)
  - **Pawn Promotion**
  - **En Passant**
- **Check detection**: Prevent illegal moves that leave the king in check.
- **Checkmate detection**: Recognize when the game ends due to checkmate.
- **Stalemate detection**: Recognize when no legal moves exist and the king is not in check.
- Move legality enforcement (players cannot make illegal moves).

4. Graphical User Interface (GUI)

A **Basic Graphical User Interface is mandatory** for this assignment.

The GUI must:

- Visually display the chessboard and all the pieces.
- Allow the human player to make moves by **typing the move in a text input box using standard algebraic notation or coordinate-based format** (e.g., e2e4 or e7 e5).
- Optionally, highlight valid moves for a selected piece.
- Display game status messages clearly (e.g., "Checkmate", "Stalemate", "AI is thinking...").
- Use any suitable GUI library such as **Tkinter**, **Pygame**, or **PyQt**.

Make sure the GUI is responsive and user-friendly to ensure smooth gameplay between the human player and the AI.

5. Code Quality & Modularity:

- Follow object-oriented programming best practices.
- Maintain clean, readable, and well-documented code.
- Separate core game logic from AI logic and GUI logic.

Evaluation Criteria:

Feature	Marks
Correct Implementation of Minimax with Alpha-Beta Pruning	40
Object-Oriented Design with Required Classes (Chess Game, Board, Piece hierarchy, Player types, etc.)	30
Proper Handling of Game Mechanics (Castling, Promotion, En Passant, Check/Checkmate, Stalemate)	30
Functional and User-Friendly GUI (board display, move input via text box, status updates)	30
PDF Report (explaining design, AI logic, testcases etc.)	20
Clean and Modular Code (well-structured, comments, PEP8-compliant)	20
Submission Format (.ipynb file, organized project structure)	10
Demo / Viva (understanding of code, ability to explain design and logic)	20
Total	200

Submission Instructions

- Assignment must be submitted on the google classroom.
- Submission other than google classroom won't not be accepted.
- You are required to submit a zip file named as **22i-7777_Name_BCY-A.zip** containing your notebook and document named as: **Asg04- <22I- XXXX>.ipynb** and **Asg04- <22I-XXXX>.pdf**.