**SPL-1 Project Report** 

**Chess Game**

Submitted by

***Md. Kamrul Hasan***

**BSSE Roll No. : 1418**

**BSSE Session: 2021-2022**

Submitted to

***Prof. Dr. Md. Shariful Islam***

***Professor***

***IIT, DU***

****

**Institute of Information Technology**

**University of Dhaka**

17-12-2023

Project Name: Chess Game

Author : Md. Kamrul Hasan

Roll - BSSE-1418

Supervised by: Prof. Dr. Shariful Islam

Professor

IIT,

University of Dhaka.

Date submitted: 17 Dec, 2023

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Signature of Prof. Dr. Shariful Islam )

### 3

### Table of Contents

1. **Introduction 4**
2. **Background of the Project 4**
3. **Description of the Project 5**
4. **Implementation and testing 6**
5. **User Interface 7**
6. **Challenges faced 9**
7. **Conclusion 9**

**References 10**

4

# Introduction

Chess is a classic board game played between two opponents on an 8x8 checkered board. Each player commands 16 pieces with unique moves. The goal is to checkmate the opponent's king, rendering it unable to escape capture.

Chess is renowned for promoting strategic thinking and tactical skill. Success involves planning, foresight, and the ability to outmaneuver the opponent. The game's universal appeal transcends cultural boundaries, providing a common ground for players worldwide.

With its rich history and enduring popularity, chess stands as a timeless pursuit that challenges the mind and fosters intellectual growth.

# Background of the Project

2.1 Chess:

Chess is a two-player strategy board game played on an 8x8 grid. It involves 16 pieces for each player, including a king, queen, bishops, knights, rooks, and pawns.

2.2 Programming Languages:

The project uses C++, a versatile and widely used programming language known for its efficiency and object-oriented features.

2.3 Algorithm Design:

Algorithms are step-by-step procedures or formulas for solving problems. In the context of chess, algorithms govern moves, validations, and game progression.

Minimax is a decision-making algorithm used in chess and other two-player games. It evaluates all possible moves to find the one that maximizes the player's advantage while minimizing the opponent's, assuming optimal play from both sides.

2.4 User Interface (UI) Design and Game Design:

UI design involves creating visually appealing and user-friendly interfaces for software applications.

5

Unicode enhances the visual appeal of the chessboard and pieces in the user interface. By using Unicode characters for chess pieces, the UI achieves an authentic and visually pleasing representation. This integration enhances the overall user experience, making the digital chess-playing environment more immersive and recognizable.

2.5 Game Logic:

2.5.1 Board Setup:

8x8 grid with 64 squares.

Each side has 16 pieces.

2.5.2 Piece Movements:

Each piece moves uniquely.

King, queen, rooks, bishops, knights, and pawns have specific ways of moving.

2.5.3 Check and Checkmate:

Check: King is under threat.

Checkmate: No legal moves to save the king.

2.5.4 Game Progression:

Players take turns making moves.

2.5.5 End Conditions:

Game ends with checkmate, stalemate, or draw.

# Description of the Project

3.1 Menu and Modes:

A menu for player vs player and player vs computer modes.

3.2 Game Logic:

Validated moves for various chess pieces using isValidMove.

Detected check and stalemate conditions.

3.3 AI Opponent:

Integrated the Minimax algorithm for the computer's move decisions.

3.4 Board Display:

Visualized the chessboard and possible moves.

6

3.5 User Interface:

Managed console interactions and user input.

3.6 Game Flow:

Established a start game loop for continuous play.

# Implementation and Testing

4.1 Main Function:

The main function initiates the game by calling the startGame function.

4.2 Game Loop (startGame function):

The startGame function contains a while loop that calls the userInterface function until the user decides to exit.

4.3 User Interface (userInterface function):

Displays a menu for the user to choose between playing against the computer, playing against another player, or exiting the game.

Calls functions like consoleSet, restartChessBoard, and restartTileBoard to initialize the game state.

Handles user input and navigates to the corresponding game mode or exits based on the user's choice.

4.4 Chessboard Initialization (restartChessBoard and restartTileBoard functions):

Initializes the chessboard with the default starting positions of pieces.

4.5 Game Modes (playerVsPlayer and playerVsComputer functions):

playerVsPlayer: Allows two players to take turns making moves on the console-based chessboard.

playerVsComputer: Enables a player to play against the computer, with a simple AI opponent.

4.6 Move Validation (isValidMove function):

Validates the legality of a move based on the rules of chess for different types of pieces.

4.7 Check and Stalemate Detection (isCheck, isStalemate functions):

Checks if a player's king is in check or if the game is in a stalemate situation.

4.8 AI Implementation (minimax, bestMove, gameMove functions):

Implements a basic AI using the minimax algorithm for the computer opponent in the playerVsComputer mode.

4.9 Minimax Algorithm:

7

The minimax algorithm is a fundamental technique in chess programming that revolves around mathematical principles to determine optimal moves in a game with alternating turns, such as chess. At its core, minimax is a decision-making algorithm employed in two-player, zero-sum games, where one player aims to minimize the possible loss, and the other seeks to maximize the potential gain.

In the context of chess, each possible move is assigned a value based on a heuristic evaluation function. The algorithm explores the game tree, representing different board positions resulting from potential moves. The depth of the tree represents the number of moves ahead the algorithm analyzes. At each level, the algorithm alternates between minimizing and maximizing values, with the minimizing player assuming the opponent makes optimal moves to minimize the score, and vice versa.

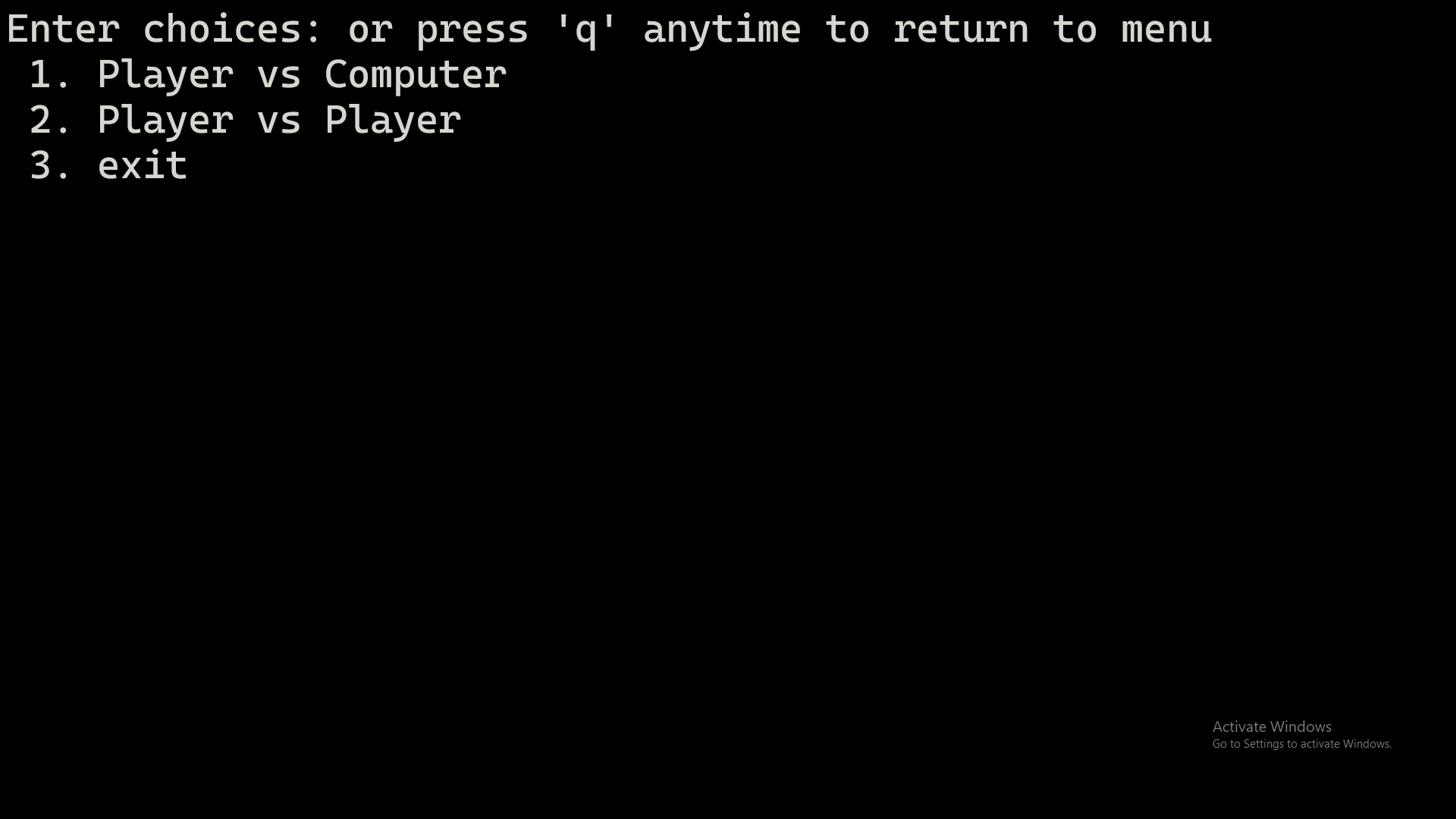
The complexity of the minimax algorithm is influenced by the branching factor (the average number of moves available at each position) and the depth of the search tree. In the worst case, the time complexity is exponential, O(n^m), where 'n' is the branching factor, and 'm' is the depth of the tree. As chess has a high branching factor, exploring all possible moves to significant depths can be computationally expensive. To mitigate this, various enhancements, such as alpha-beta pruning, are employed to reduce the number of evaluated nodes and improve the algorithm's efficiency.

In summary, the minimax algorithm in chess employs mathematical principles to traverse the game tree, assigning values to moves and determining optimal strategies through a recursive approach. However, its computational complexity necessitates the use of optimization techniques to make it viable for real-world chess engines.

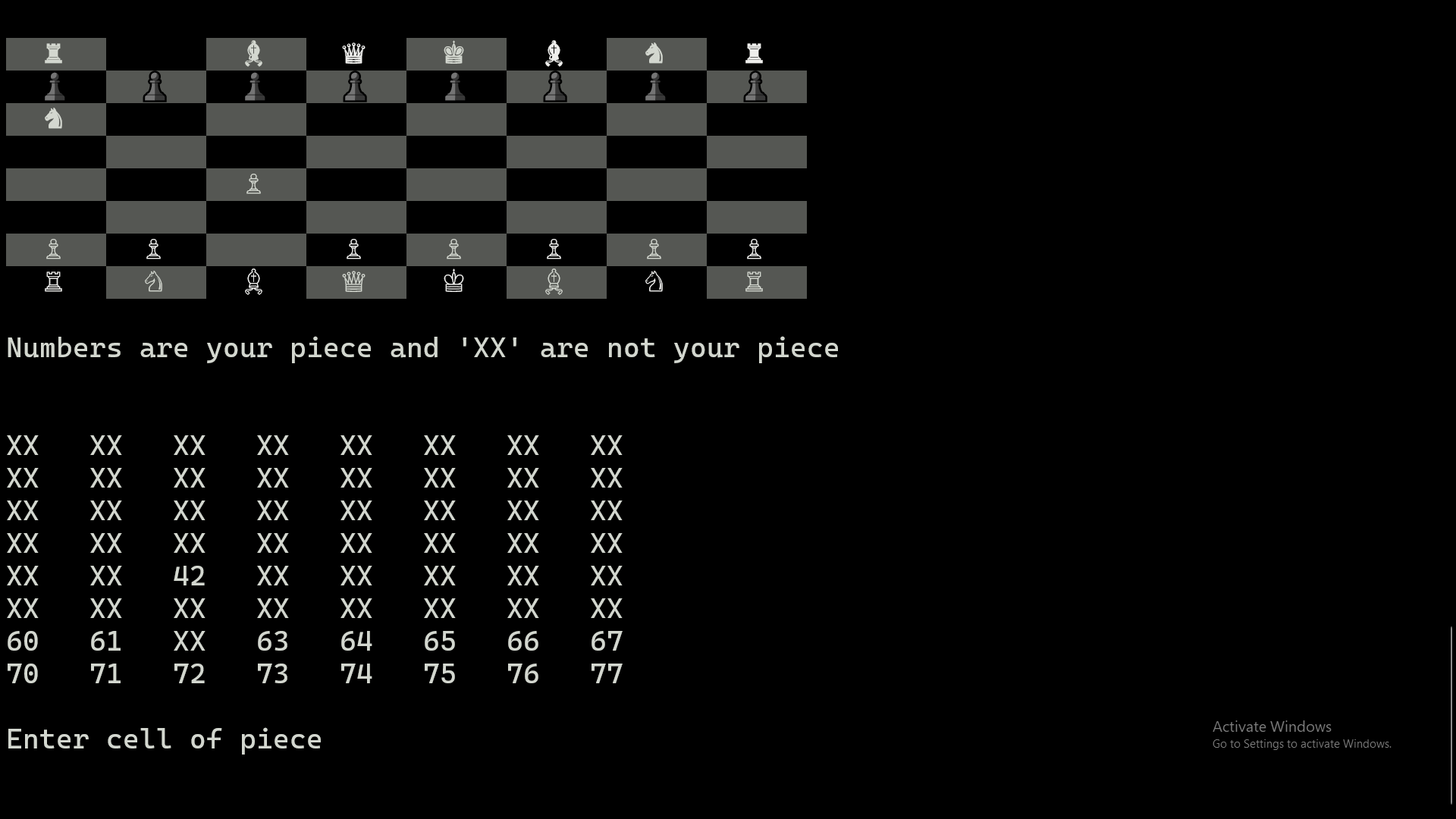
# User Interface

A terminal based user interface

8



Unicode based chess pieces



9

# Challenges Faced

6.1 Complex Game Logic:

Challenge: Chess involves intricate rules for each piece and special moves like castling.

Solution: Careful implementation of the rules and thorough testing to ensure the game follows standard chess logic.

6.2 User Input Handling:

Challenge: Processing and validating user input for moves can be error-prone.

Solution: Implement input handling (string type input )and error-checking (try - catch block )mechanisms to ensure valid moves.

6.3 AI Implementation:

Challenge: Designing a competent AI opponent with various difficulty levels.

Solution: Implementing minimax.

6.4 Board and piece:

Challenge: Representing board and pieces without graphics tool.

Solution: Using Colored texts for board representation and using unicode for piece representation.

# Conclusion

7.1 Things I have learned from the project–

In the realm of chess programming, algorithmic thinking plays a pivotal role in crafting the game logic, move generation, and the implementation of artificial intelligence (AI) algorithms. Delving into AI implementation for chess offers valuable insights into algorithms like minimax and heuristic evaluation functions, crucial for predicting optimal moves and determining winning strategies. The design of a chess interface becomes a lesson in user experience (UX) design principles, emphasizing clarity, responsiveness, and player engagement. Given the complexity of chess scenarios and edge cases, robust error handling and code development become imperative for creating a reliable program. Furthermore, the intricacies of chess contribute to understanding game theory, shedding light on strategic decision-making and the pursuit of optimal moves.

7.2 Things that could be done in the future–

10

In the quest for enhancing the chess game, the plan encompasses implementing more advanced AI algorithms, possibly leveraging machine learning techniques. An online multiplayer feature is proposed, facilitating internet-based matches with a matchmaking system for balanced competition. Graphical improvements aim to enhance visual aesthetics with realistic designs, smooth animations, and customizable themes. Rule variations, such as specific conditions for castling and the introduction of new pieces, are considered to add strategic depth. Together, these upgrades aim to elevate the overall chess gaming experience through improved AI, online multiplayer, enhanced visuals, and intriguing rule variations.

**Reference**

1. Fast Modulo function – <https://stackoverflow.com/questions/33333363/built-in-mod-vs-custom-mod-function-improve-the-performance-of-modulus-op> , stackoverflow, 10-12-2023

[2] Chess pieces unicode symbols –

<https://en.wikipedia.org/wiki/Chess_symbols_in_Unicode> , Wikipedia, 10-12-2023

[3] Print color texts –

<https://iq.opengenus.org/print-text-in-color-in-c/> , Open Genus, 10-12-2023

[4] Minimax Algorithm –  
 <https://www.youtube.com/watch?v=l-hh51ncgDI&ab_channel=SebastianLague> ,

Youtube , 10-12-203

[5] Enter detection in C/C++ –

<https://stackoverflow.com/questions/42818899/detecting-enter-key-in-c> , stackoverflow , 10 -12-2023