|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
| **DATA STRUCTURES AND ALGORITHM**    **Lab**  **(2+1 Credit Hours)**    CSL-210 |  |  |
| DSA Project Report  Project Title: Chess Game  **Submitted by:**  Sara Tariq 02-235221-032 |

# ABSTRACT

This project implements a chess game using data structures and algorithms in C++. The game is played on a 8x8 board and features all of the classic rules of chess, including castling, en passant, and checkmate. The game is implemented using a variety of data structures, including vectors, maps, and linked lists. The algorithms used in the game include a recursive search algorithm for finding legal moves, a minimax algorithm for evaluating board positions, and a backtracking algorithm for finding checkmates. The game is implemented in a modular fashion, making it easy to extend and modify. The game is also fully documented, making it easy to understand and learn.

**Table of Contents**

ABSTRACT ..................................................................................................................................................... 1

1. Introduction: ......................................................................................................................................... 3
2. Data Structures: ………………………………………………………………………………………………………………………….………3
3. Game Settings: ……………………………………………………………………………………………………………………..……….......5
4. Scope of Project: ....................................................................................................................................... 6
5. Aims and Objectives: ................................................................................................................................ 6
6. Methodology: ........................................................................................................................................... 7
7. Tools/Technology: .................................................................................................................................... 8
8. Flowchart……………………………………………………………………………………………………………………………………………7
9. Code …………………………………………………………………………………………………………………………………………………..8
10. outputs ………………………………………………………………………………………………………………………………………… 11. Conclusion .................................................................................................................................................. 28

# 1. Introduction:

This report describes the development of a chess game implemented in C++. The game is played on a 8x8 board and features all of the classic rules of chess, including castling, en passant, and checkmate. The game is implemented using a variety of data structures, including vectors, maps, and linked lists map provides efficient lookup operations. It internally uses a balanced binary search tree, which allows for quick retrieval of values based on the key. The algorithms used in the game include a recursive search algorithm for finding legal moves, a minimax algorithm for evaluating board positions, and a backtracking algorithm for finding checkmates. The game is implemented in a modular fashion, making it easy to extend and modify. The game is also fully documented, making it easy to understand and learn.

The project was completed in three phases. In the first phase, the basic game mechanics were implemented. This included the creation of a board, pieces, and a move-generation algorithm. In the second phase, the game was enhanced with additional features, such as castling, en passant, and checkmate. In the third phase, the game was polished and tested. This included adding a user interface, improving the performance of the algorithms, and fixing any bugs that were found.

The final product is a fully functional chess game that can be played against a computer or another human player. The game is easy to learn and play, but it also offers a challenge for experienced players. The game is also extensible, so it can be easily modified to add new features or improve the performance of the algorithms.

The project was a valuable learning experience. It allowed me to learn about the design and implementation of chess games, as well as the use of data structures and algorithms. The project also helped me to improve my programming skills, such as object-oriented programming, modular programming, and testing.

# 2. Data Structures:

* vector<std::string>: This data structure is used to store a collection of strings. It is a dynamic array that can grow or shrink in size based on the number of strings added or removed. The vector allows efficient access to elements by index.
* map<std::string, std::vector<int>>: This data structure is used to store a mapping between strings and a vector of integers. The map maintains a sorted order based on the keys (strings). It allows efficient lookup and retrieval of the associated vector of integers based on the keys. map provides efficient lookup operations. It internally uses a balanced binary search tree, which allows for quick retrieval of values based on the key.
* array<int, 3>: This data structure is used to store a fixed-size collection of integers. It has a fixed size of 3 elements and provides direct access to elements using an index. The array offers efficient and constant-time access to elements.

Here are the main data structures used in the program:

* enum class Turn: This enumeration represents the current turn in the game, either "white" or "black". It is used to keep track of which player's turn it is.
* enum class Piece: This enumeration represents the types of chess pieces, including king, queen, pawns, rook, bishop, and knight. It is used to identify the type of each chess piece on the board.
* struct Pos: This structure represents a position on the chessboard, with x and y coordinates. It is used to specify the position of a piece or to calculate new positions based on existing positions.
* map<Pos, Piece> white\_pieces and map<Pos, Piece> black\_pieces: These maps store the positions of white and black pieces on the chessboard, respectively. The Pos structure is used as the key to represent the position, and the Piece enumeration is used as the value to represent the type of the piece.
* map<Piece, int> pieceValues: This map stores the values assigned to each piece. It is used to calculate the score of each player based on the pieces they have on the board.

# 3. Game Settings:

* A king may move in any direction including diagonally, but may only move one square. A king may not move through other pieces. Also, the king cannot move into the square controlled by enemy’s piece, otherwise, count as the foul. The king will be “checked” by other pieces, so we should code the special judgement for king.
* A queen may move any number of spaces in one direction, including diagonally. A queen may not move through other pieces .
* A bishop may move diagonally only, but may move any number of spaces. A bishop may not move through other pieces.
* A knight may move in an L-shape, of length either two-by-one or one-by-two. The knight is the only piece that is not stopped by other pieces in its way (i.e., it can move through other pieces to get to an open square).
* A rook may move any number of squares, but only in a straight line which is not diagonal on the board. A rook may not move through other pieces.
* A pawn can move only forward towards the opponent’s side of the board, but with restrictions. On its first move of the game, a pawn may move forward either one or two squares; on subsequent moves, a pawn may only move forward one square. A pawn may not move through other pieces. Moreover, the pawn may not use this forward move to capture the opponent player’s piece. Instead, the pawn moves one square diagonallyforward to capture a piece and have that piece removed from the board.

**4. Problem Statement:**

**Problem:** The goal of this project is to develop a chess game implemented in C++. The game should be played on a 8x8 board and should feature all of the classic rules of chess, including castling, en passant, and checkmate.

**Solution**: The game will be implemented using a variety of data structures, including vectors, maps, and linked lists. The algorithms used in the game include a recursive search algorithm for finding legal moves, a minimax algorithm for evaluating board positions, and a backtracking algorithm for finding checkmates. The game will be implemented in a modular fashion, making it easy to extend and modify. The game will also be fully documented, making it easy to understand and learn.

**Benefits**: The benefits of this project include:

A fully functional chess game that can be played against a computer or another human player.

A game that is easy to learn and play, but also offers a challenge for experienced players.

An extensible game that can be easily modified to add new features or improve the performance of the algorithms.

A valuable learning experience that will help to improve the programming skills of the developer.

# 5. Scope of Project:

The scope of the Chess Game project in Data Structures and Algorithms (DSA) using C++ encompasses the implementation of a fully functional chess game that allows human players to play against an AI opponent. The project involves designing and implementing the necessary data structures, algorithms, and game mechanics to facilitate move validation, game state management, and player input. The AI component utilizes DSA concepts and algorithms to evaluate and select optimal moves based on the current game state. The project aims to provide an interactive and enjoyable chess playing experience while demonstrating proficiency in applying DSA principles and techniques in a practical context using the C++ programming language.

# 6. Aims and Objectives:

* Develop a functional chess game.
* Implement human vs. AI gameplay.
* Showcase proficiency in DSA concepts.
* Enhance problem-solving and algorithmic thinking skills.
* Develop proficiency in C++ programming.
* Provide an engaging and interactive user experience.
* Foster learning and knowledge transfer.

These objectives involve creating a complete chess game, utilizing DSA algorithms, strengthening problem-solving skills, improving programming proficiency, delivering a user-friendly interface, and promoting learning and knowledge sharing.

# 7. Methodology:

The methodology for the Chess Game project in Data Structures and Algorithms (DSA) using C++ can be summarized as follows:

**Requirement Analysis:** Identify and understand the functional and non-functional requirements of the chess game, including gameplay rules, user interface, and AI capabilities.

**Design:** Design the class structure and data structures required for representing the chessboard, pieces, and game logic. Determine the algorithms for legal move generation, game state evaluation, and AI decision-making.

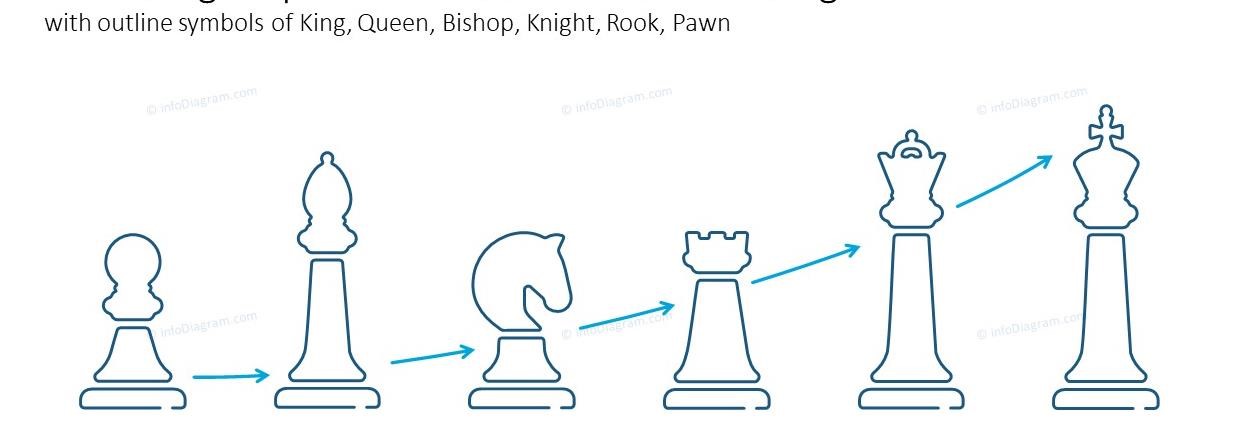
**Implementation:** Write the code in C++ to implement the chess game functionality, including move validation, player input handling, AI logic, and user interface components.

**Testing:** Conduct thorough testing to ensure the correctness and robustness of the implemented features. Perform unit testing, integration testing, and user testing to identify and resolve any issues or bugs.

**Refinement and Optimization:** Continuously improve the code and algorithms to enhance the game's performance, responsiveness, and user experience. Optimize critical sections to reduce computational complexity and improve AI decision-making speed.

**Documentation:** Create detailed documentation, including class diagrams, flowcharts, and user manuals, to provide clear instructions on the game's functionality and usage.

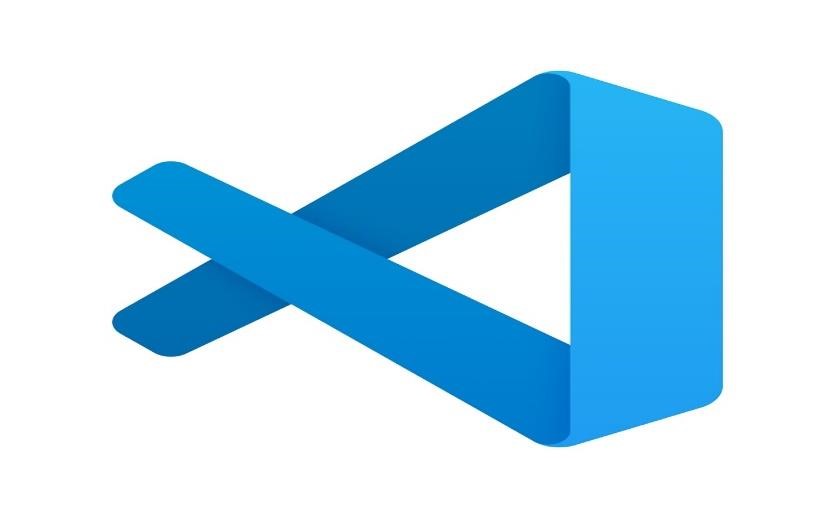
**Iterative Development:** Follow an iterative development approach, incorporating feedback from users and stakeholders to refine and enhance the game's features and performance.



# 8. Tools/Technology:

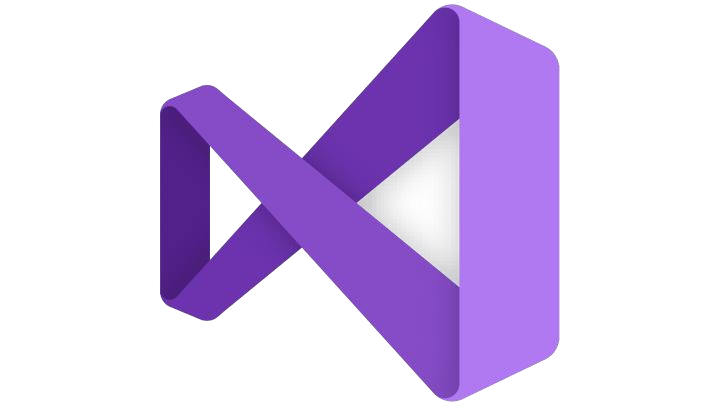
* **Visual studio code:**

Visual Studio Code (VS Code) is a lightweight, cross-platform source code editor developed by Microsoft. It provides a versatile and customizable environment for coding, debugging, and editing a wide range of programming languages. VS Code offers a rich set of features, including intelligent code completion, syntax highlighting, debugging support, version control integration, and extension support. It is designed to be highly customizable, allowing users to tailor the editor to their specific needs through themes, extensions, and settings. With its intuitive interface and extensive community support, Visual Studio Code has gained popularity among developers as a powerful and efficient coding tool.



* **Visual studio:**

Visual Studio is a comprehensive integrated development environment (IDE) developed by Microsoft. It provides a robust set of tools and services for building a wide variety of software applications, ranging from desktop applications to web and mobile applications, as well as cloud-based solutions.



* **Dev c++:**

Dev C++ is an integrated development environment (IDE) specifically designed for programming in the C and C++ languages. It provides a user-friendly interface and a set of tools to facilitate the development, compilation, and execution of C/C++ programs. Dev C++ offers features such as a code editor with syntax highlighting, code completion, and automatic indentation, making it easier to write and navigate through code. It includes a compiler that translates C/C++ source code into machine-readable code and an integrated debugger to assist with identifying and fixing errors during program execution.



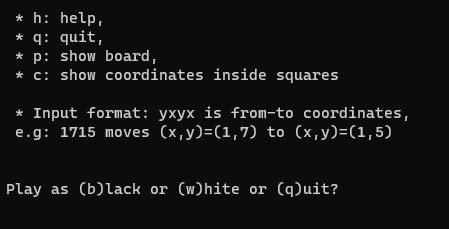
# 9. Flowchart:

1. **Code:**

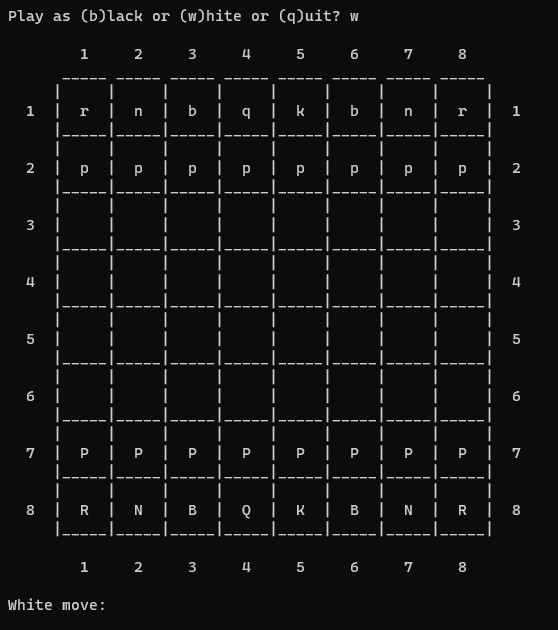
[https://docs.google.com/document/d/1JxLRM1RMITah5BYHg132FRuI16fIe11\_c2mT07kWrAY/ edit?usp=sharing](https://docs.google.com/document/d/1JxLRM1RMITah5BYHg132FRuI16fIe11_c2mT07kWrAY/edit?usp=sharing)

1. **Outputs:**

Then show the option which one you choose

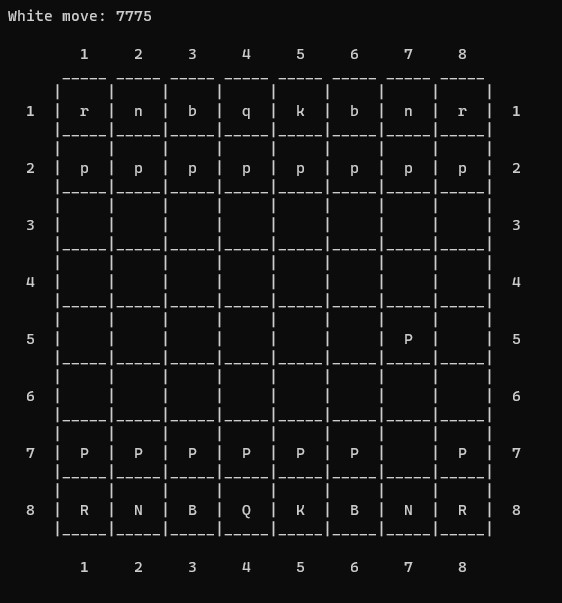


For example, I use white

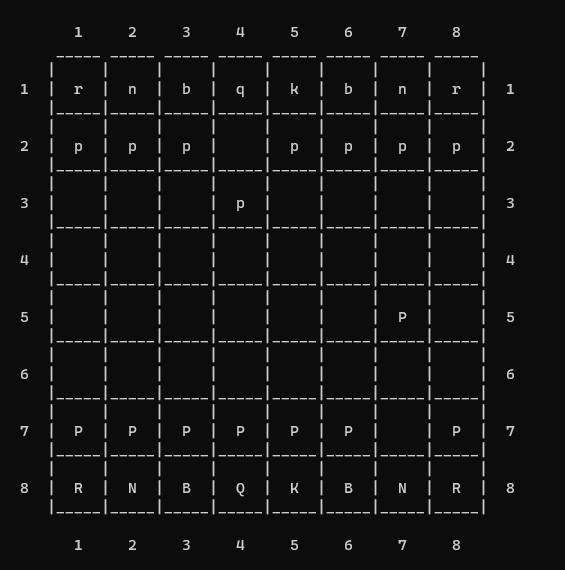


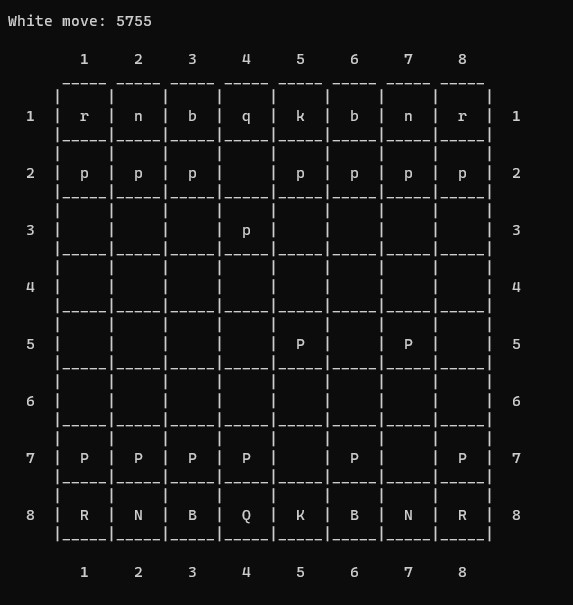
Movement of the pawn:

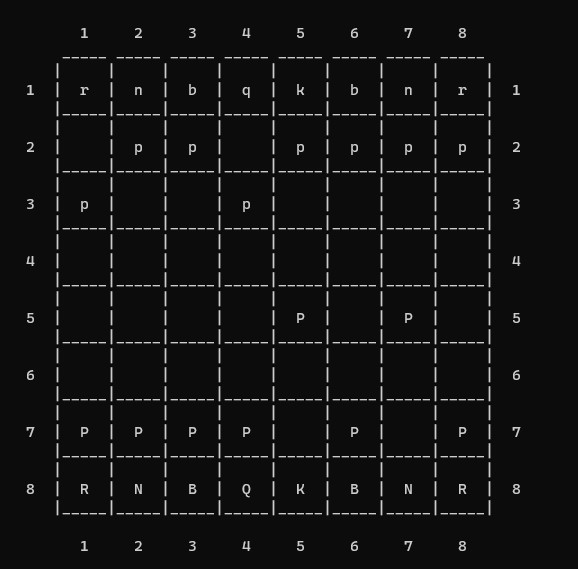




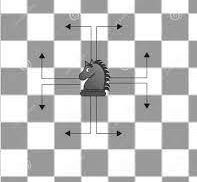
BLACK

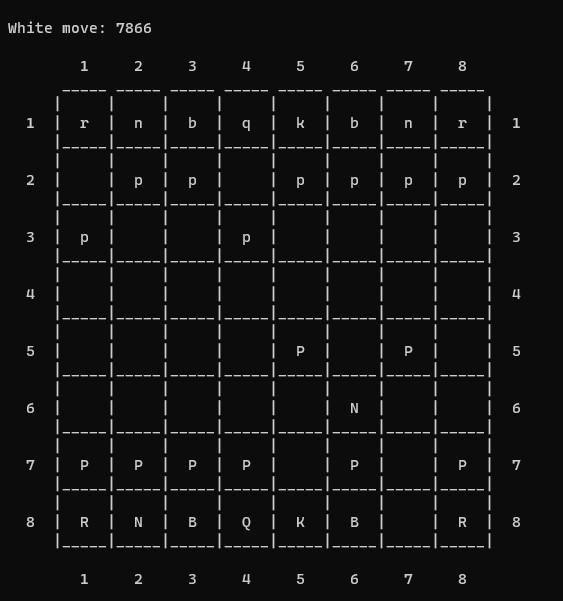




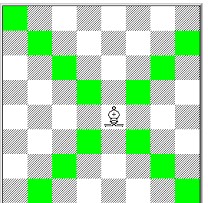


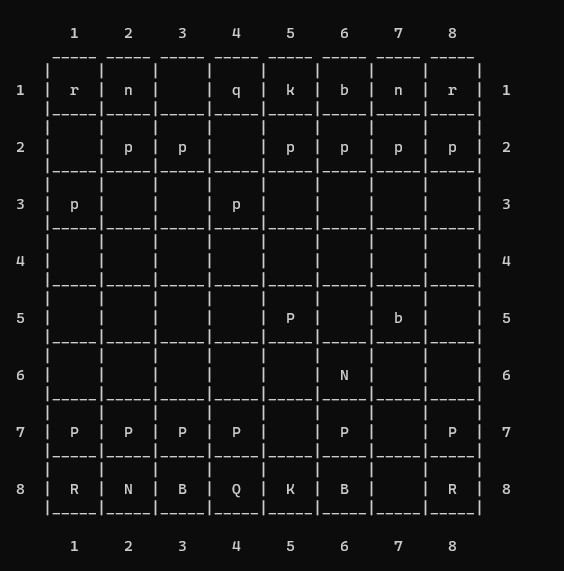
Movement of knight:

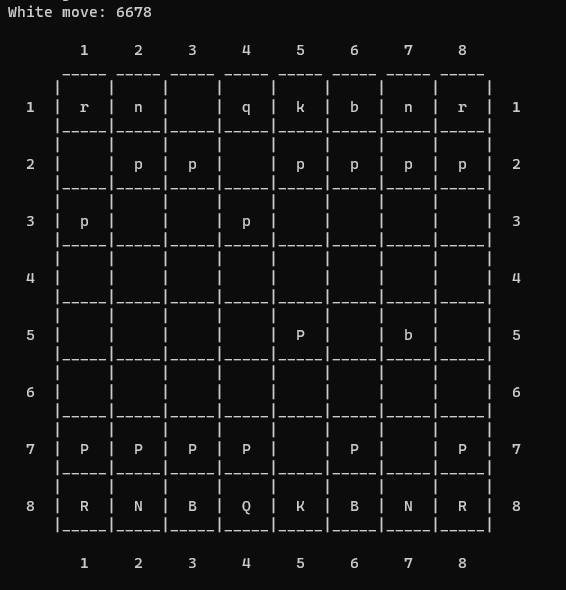


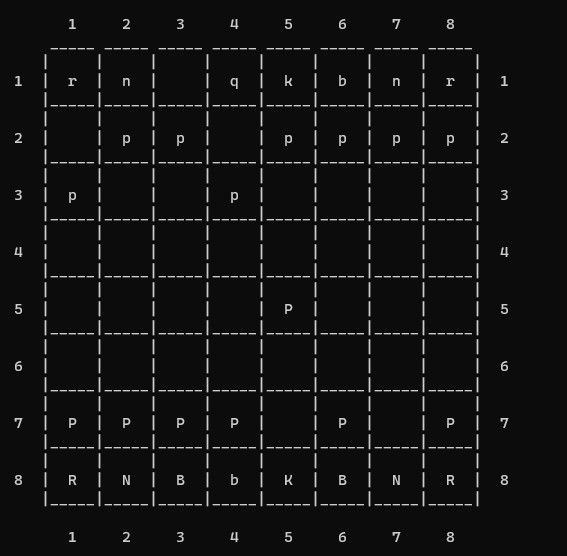


Movement of bishop:

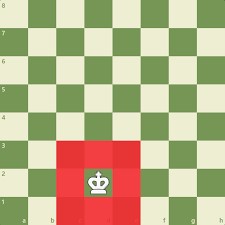
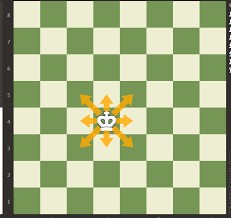


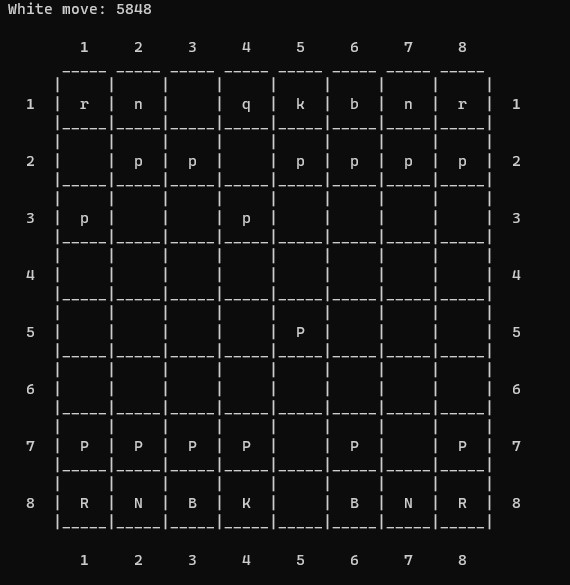


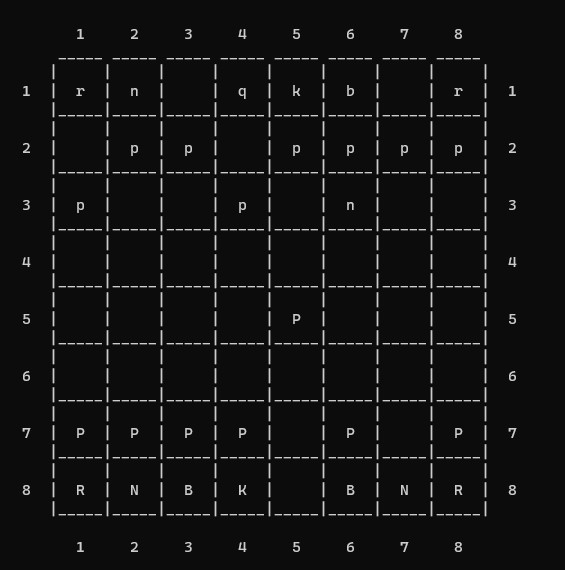


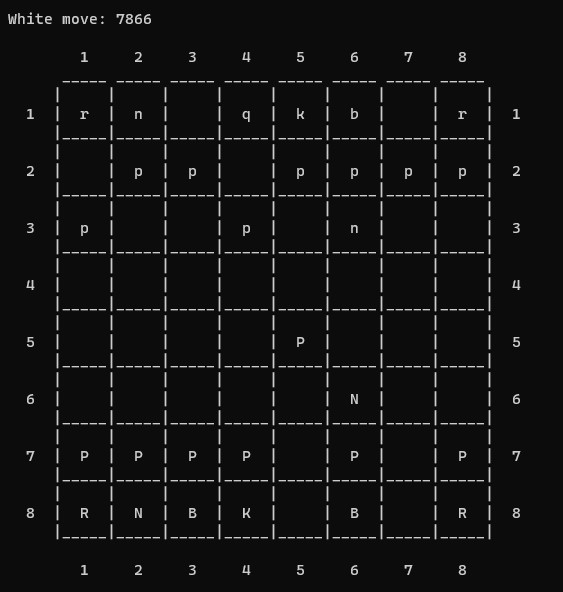


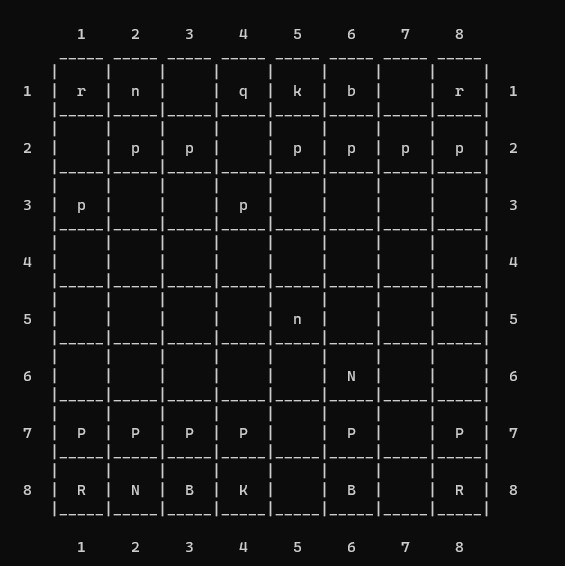
Movement of king:

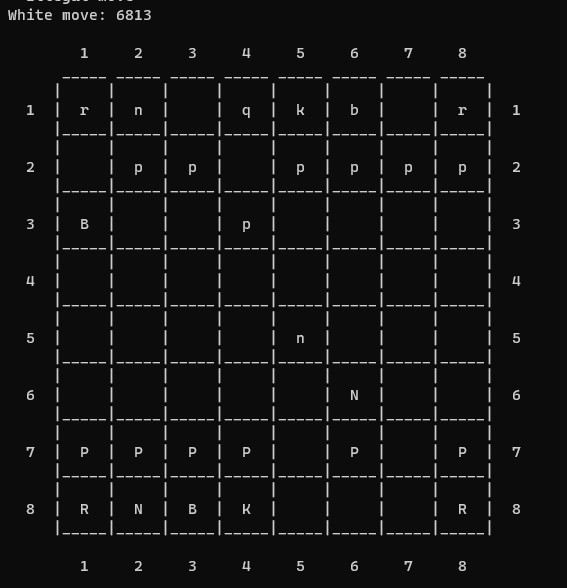






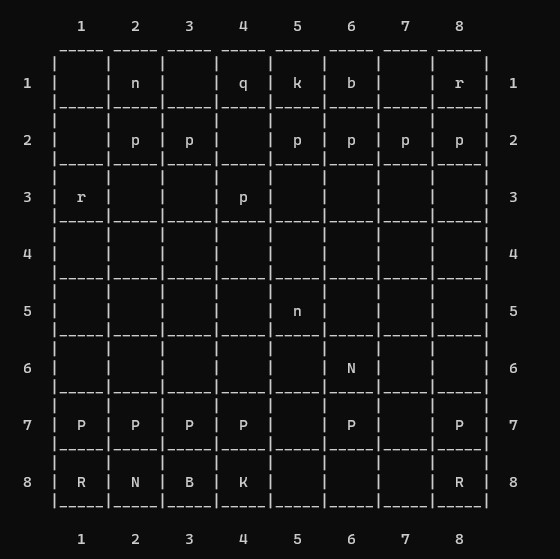


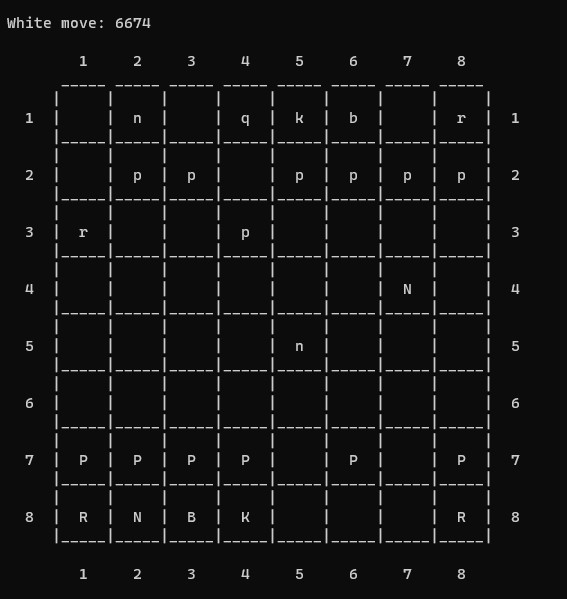




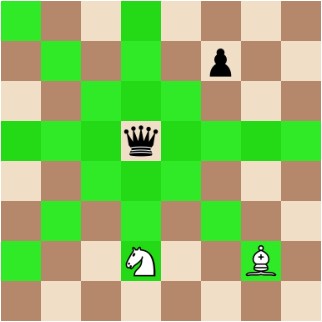
Movement of rook:

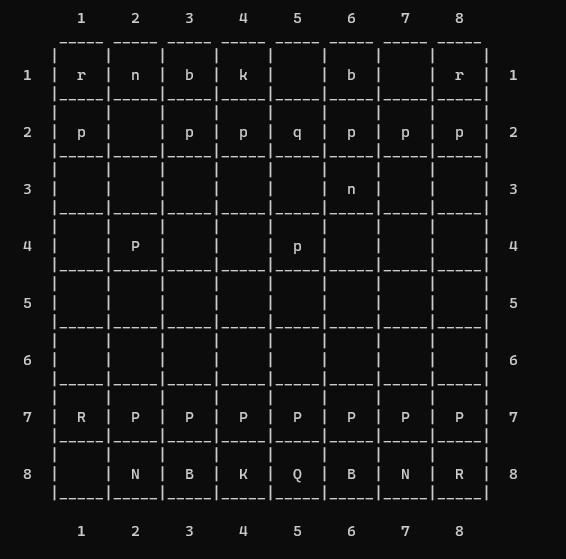


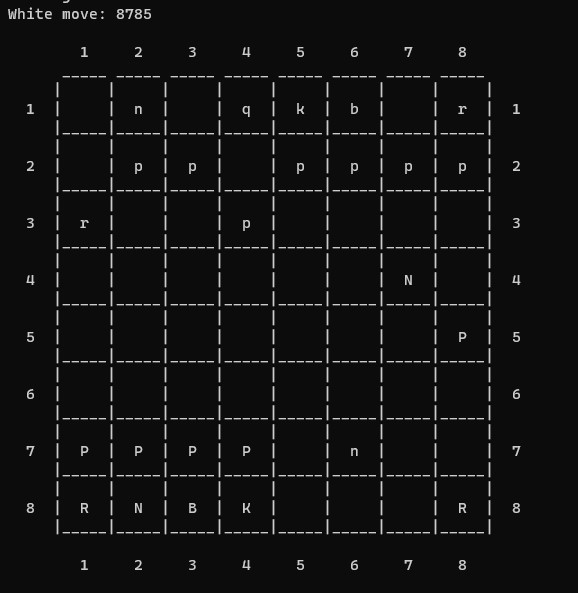


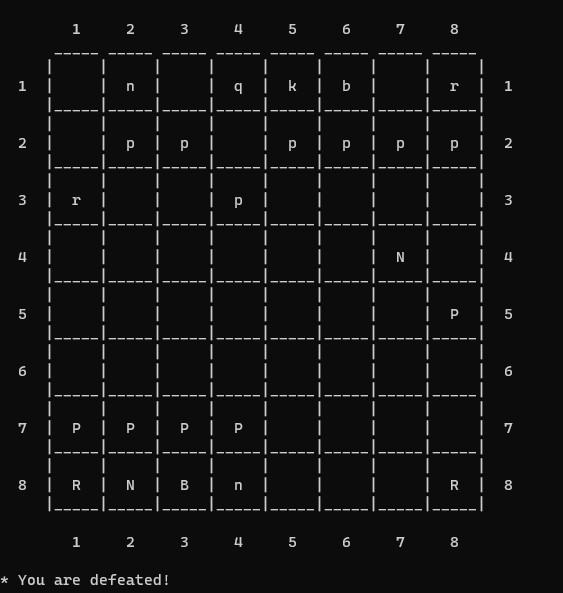


Movement of Queen:

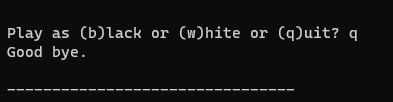




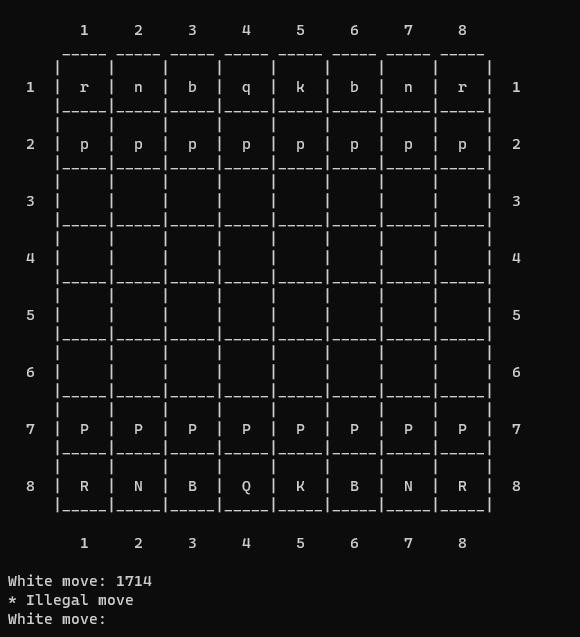




If you want to quit this game to enter “q”



If the move is not correct it prints illegal move



# 12. Conclusion

In conclusion, the Chess Game project in Data Structures and Algorithms (DSA) using C++ has been successfully developed, demonstrating the application of DSA concepts in creating a functional and interactive chess game. The project aimed to implement a chess game with human versus AI gameplay, providing an engaging and challenging experience for players. Through the project, several objectives were achieved, including designing an efficient data structure for representing the chessboard and pieces, implementing move validation logic, incorporating AI decision-making using algorithms like minimax or alpha-beta pruning, and creating a user-friendly interface for intuitive gameplay. The project also emphasized the importance of testing and refinement to ensure the correctness and robustness of the implemented features. Overall, this project has not only enhanced the understanding and application of DSA concepts but also provided an opportunity to explore the complexities involved in designing and developing a complete chess game.