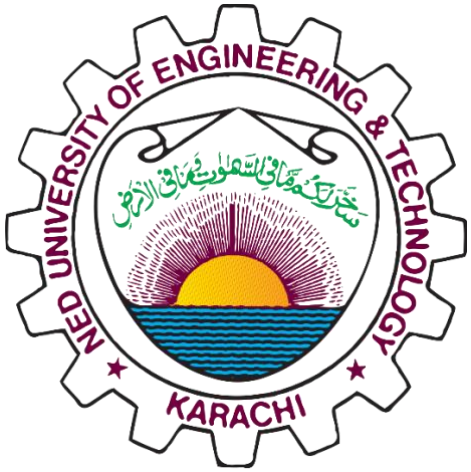
NED UNIVERSITY OF ENGINEERING & TECHNOLOGY

COMPUTER & INFORMATION SYSTEM ENGINEERING DEPARTMENT

PROJECT REPORT



CHECKERS GAME

COURSE CODE	CS-218
GROUP MEMBERS	SYED MALLAHIM ALI **CS-056** YUSRA HASSAN **CS-078**
LAB GROUP	G3
SUBMITTED TO	MS. IBSHAR ISHRAT

PROBLEM DESCRIPTION:

Our game is a checkers game implemented in Python using the Pygame library. The game uses the minimax algorithm to determine the best move for the computer player. The game's board is represented by a 2D array, and each piece on the board is represented by an object that stores its position and color. The game includes a GUI that displays the board and pieces, and allows the user to interact with the game through mouse clicks. The game also includes a function to draw all valid moves for a selected piece, and a function to simulate the effects of a move on the board. The game also has a function that uses the minimax algorithm to determine the best move for the computer player, and a function that updates the game's state and redraws the GUI. The game also has a function that check for the winner of the game.

DISTINGUISHING FEATURES:

- The game uses the Minimax algorithm to determine the computer's moves, which allows it to make strategic decisions during gameplay.
- The game uses the Pygame library to handle graphics and user input, which allows for an interactive and visually pleasing experience.
- The game includes the ability to jump over and capture opponent pieces, adding an additional layer of strategy to the gameplay.
- The game also includes the ability for the computer to make moves, allowing for a single player experience.
- The game also include the ability for the player to select a piece and the game will highlight the valid moves for that piece.
- The game uses deepcopy() to create a new copy of the board for each possible move, allowing for efficient simulation of potential moves without modifying the actual game state.

FLOW OF GAME:

The game starts with the initialization of the checkers board and the assets using the pygame library .In the main loop of the game, the player controls the blue pieces and the computer controls the white pieces. The game begins with the player making a move by clicking on one of their pieces and then clicking on the square they want to move it to. The computer will then make a move using the minimax algorithm to determine the best move. The game continues in this way, with the player and computer taking turns making moves, until one player wins by capturing all of the opponent's pieces or the game ends in a draw.. After the move is made, the game checks if there is a winner, and if there is, the game ends and the winner is printed.

If it is not the turn of the computer, the game waits for the player to click on a piece and select it. Then waits for the player to click on a valid move and makes the move. Finally, the game updates the display and waits for the next event. The game continues until the player quits or a winner is determined.

CONTROL:

In terms of control keys, the game is controlled entirely by the mouse. The player clicks on a piece to select it and then clicks on the square they want to move it to. There is no need to use any keyboard controls in this game.

DATA STRUCTURE USED:

The above game is a checkers game that uses the minimax algorithm to generate the computer's moves. The game is played on a standard 8x8 checkers board and uses the standard rules of checkers. The game's board is represented by a 2D array, and each piece on the board is represented by an object that stores its position and color.

The algorithm uses a recursive function to evaluate the potential moves .The algorithm returns the best move and the resulting board position, which is then used by the game to make the move.

INDIVIDUAL CONTRIBUTIONS:

SYED MALLAHIM ALI was responsible for implementing data structures and user interface. He worked on board.py, game.py, algorithm.py files.

YUSRA HASSAN made project report and worked on piece.py and constant.py files.

FUTURE EXPANSIONS:

These are some of the possible features that could be added in the future to improve the game and make it more engaging for the players:

- Different difficulty levels for the computer player
- Use of different evaluation functions to evaluate the board state
- Multiplayer support over a network
- Adding more advanced move types such as "flying king"
- Implementing AI neural network rather than Minimax algorithm
- Allow player to customize the game board, pieces and colors.
- These are some of the possible features that could be added in the future to improve the game and make it more engaging for the players.

TECHNOLOGY USED:

We used Pygame library to build this game other than that minimax algorithm is also used in this game.

CONCLUSION:

The team was able to successfully develop an engaging game using data structure & algorithms which was a bit challenging but rewarding as well .it polished our skills in python, problem solving and game development.

OUTPUTS: