***CHECKERS USING MINIMAX ALGORITHM***

***-Kshitij Shah(2017140048)***

***-Tirth Parekh(2017140037)***

1. How does Checkers work?

**Checkers** is a group of [strategy](https://en.wikipedia.org/wiki/Abstract_strategy_game) [board games](https://en.wikipedia.org/wiki/Board_game) for two players which involve [diagonal](https://en.wikipedia.org/wiki/Diagonal) moves of uniform game pieces and mandatory captures by jumping over opponent pieces

Checkers is played by two opponents, on opposite sides of the gameboard. One player has the dark pieces; the other has the light pieces. Players alternate turns. A player may not move an opponent's piece. A move consists of moving a piece diagonally to an adjacent unoccupied square. If the adjacent square contains an opponent's piece, and the square immediately beyond it is vacant, the piece may be captured (and removed from the game) by jumping over it.

Only the dark squares of the checkered board are used. A piece may move only diagonally into an unoccupied square. Capturing is mandatory. In almost all variants, the player without pieces remaining, or who cannot move due to being blocked, loses the game.

Uncrowned pieces move one step diagonally forwards, and capture an opponent's piece by moving two consecutive steps in the same line, jumping over the piece on the first step.

When a piece reaches the *kings row*  it becomes a *king*, and is marked by placing an additional piece on top of the first man, and acquires additional powers including the ability to move backwards and capture backwards.

1. What is minimax algorithm?

Minimax is a kind of [backtracking](https://www.geeksforgeeks.org/tag/backtracking/) algorithm that is used in decision making and game theory to find the optimal move for a player, assuming that your opponent also plays optimally.

In Minimax the two players are called maximizer and minimizer. The **maximizer** tries to get the highest score possible while the **minimizer** tries to do the opposite and get the lowest score possible.

Every board state has a value associated with it. In a given state if the maximizer has upper hand then, the score of the board will tend to be some positive value. If the minimizer has the upper hand in that board state then it will tend to be some negative value. The values of the board are calculated by some heuristics which are unique for every type of game.

A black and red text

Description automatically generated

1. Working:-

Every checker piece have the same attribute hence we

have created a class which have two attributes color and king. All the checker pieces are objects of the class. A 2D array of checker objects are created which holds the position of all the checker pieces. Methods have been written for calculating available jump and regular moves. GUI is done using pygame which is a built in a library in python.

For minimax algorithm to work every move has been given a heuristic values. In our game we have given the following heuristic values:-

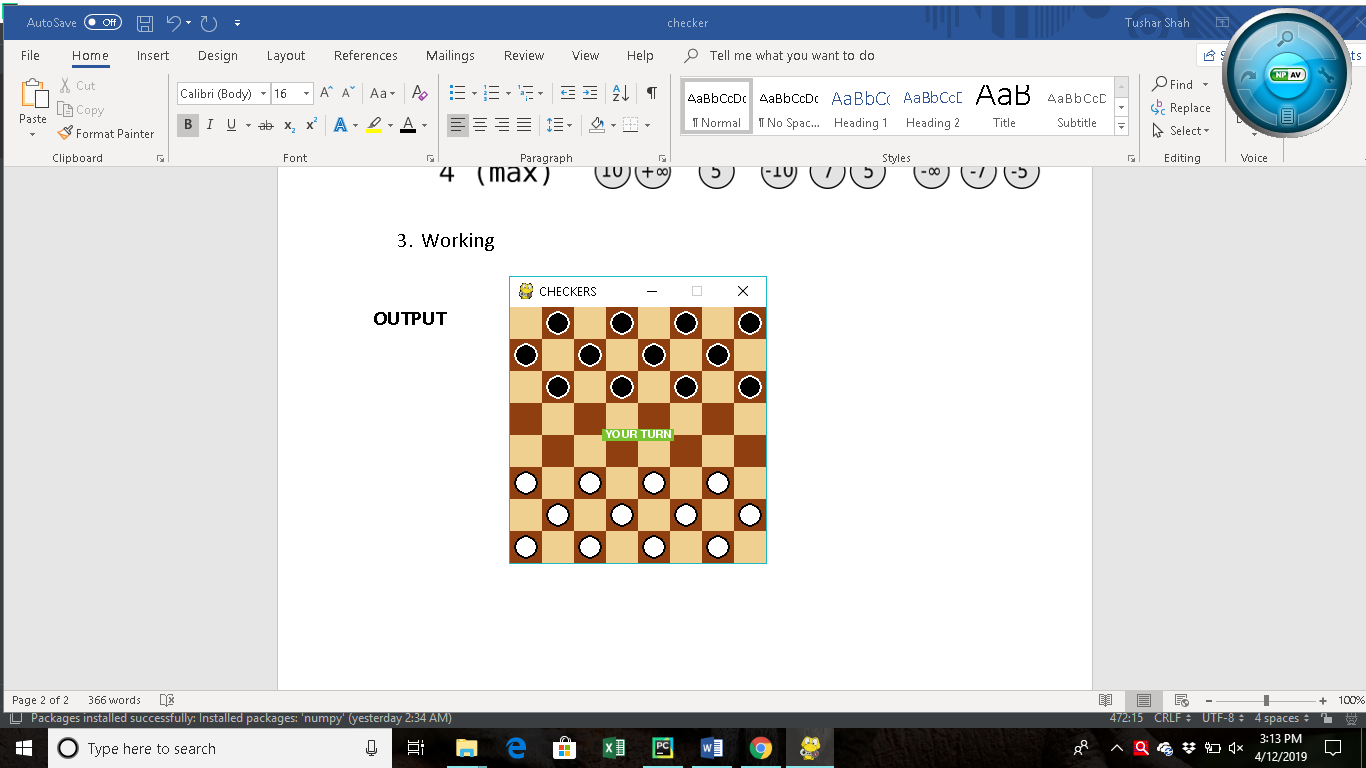
* Every checker piece has been given a value of 100
* Every checker piece which is a king has been given a value of 175
* Every checker piece going in the opposite side are given a bonus. The higher the piece goes into the opposing half higher is the bonus given the checker piece
* A king in the corner is a disadvantage and hence 25 points are deducted from the value

In the end the heuristic value of each checker piece is calculated depending on the above values. In minimax one tries to maximize the value while the other player tries to minimize the value. A depth of 4 has been set as it impossible to get all the possible moves, hence our algorithm only searches moves upto a depth of 4 and returns the best possible move.

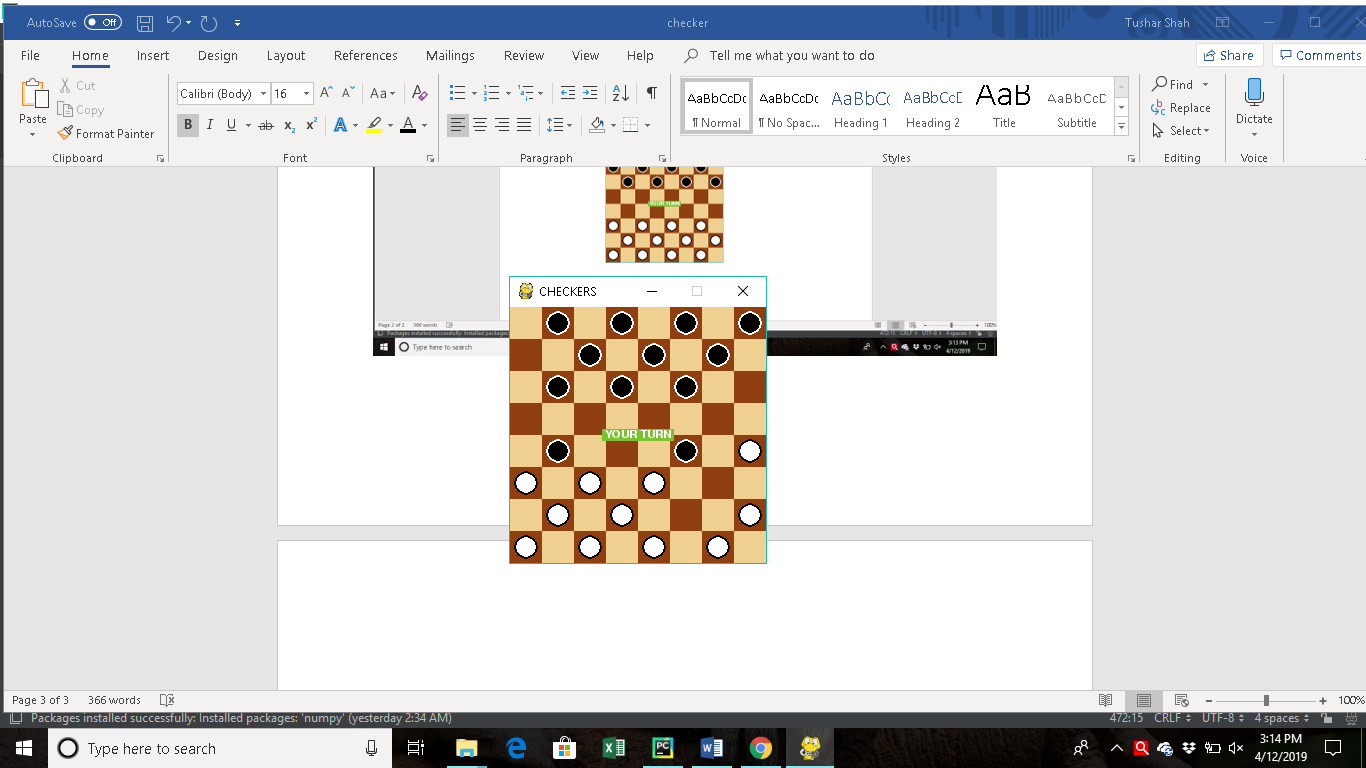
The game ends when either of the black or white pieces get over and the remaining pieces are declared as winner or no moves are left and the game is called a draw.

**OUTPUT**

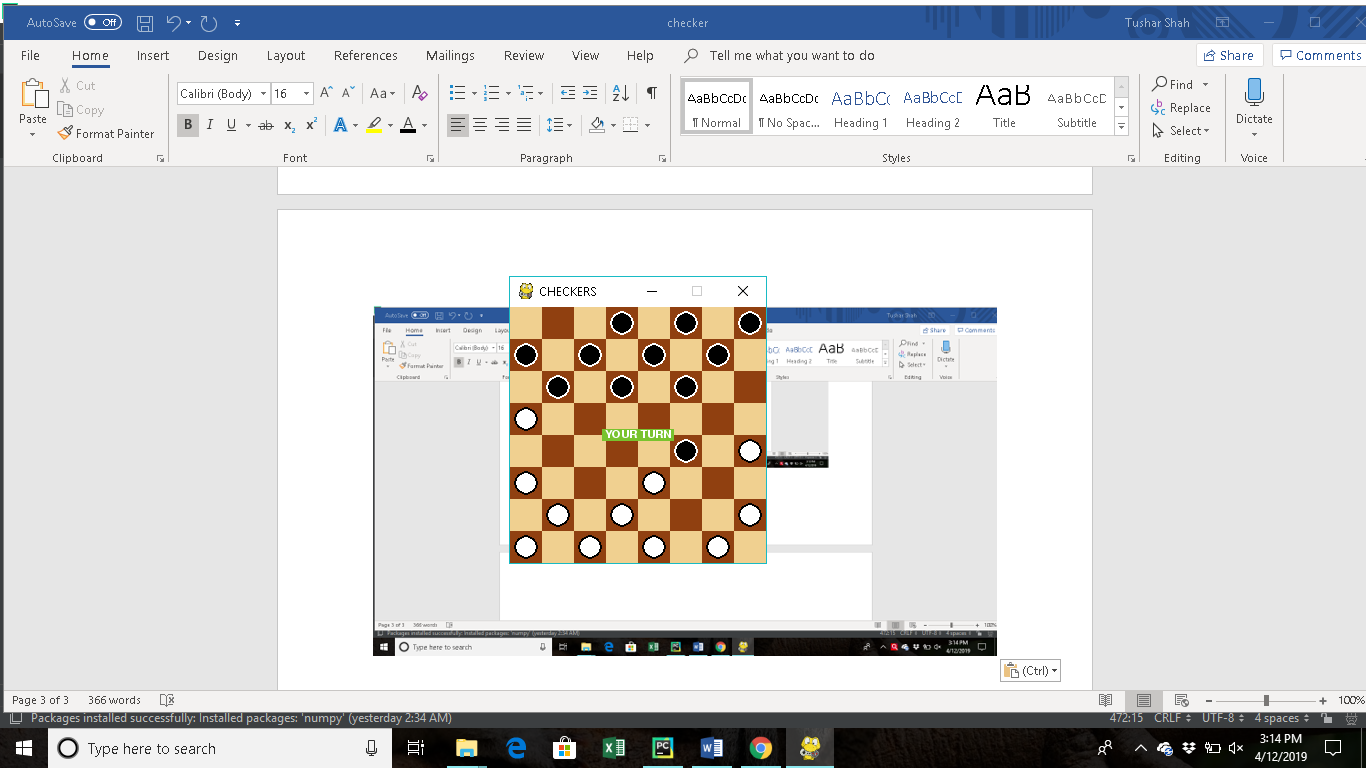
**1.**Start of the game



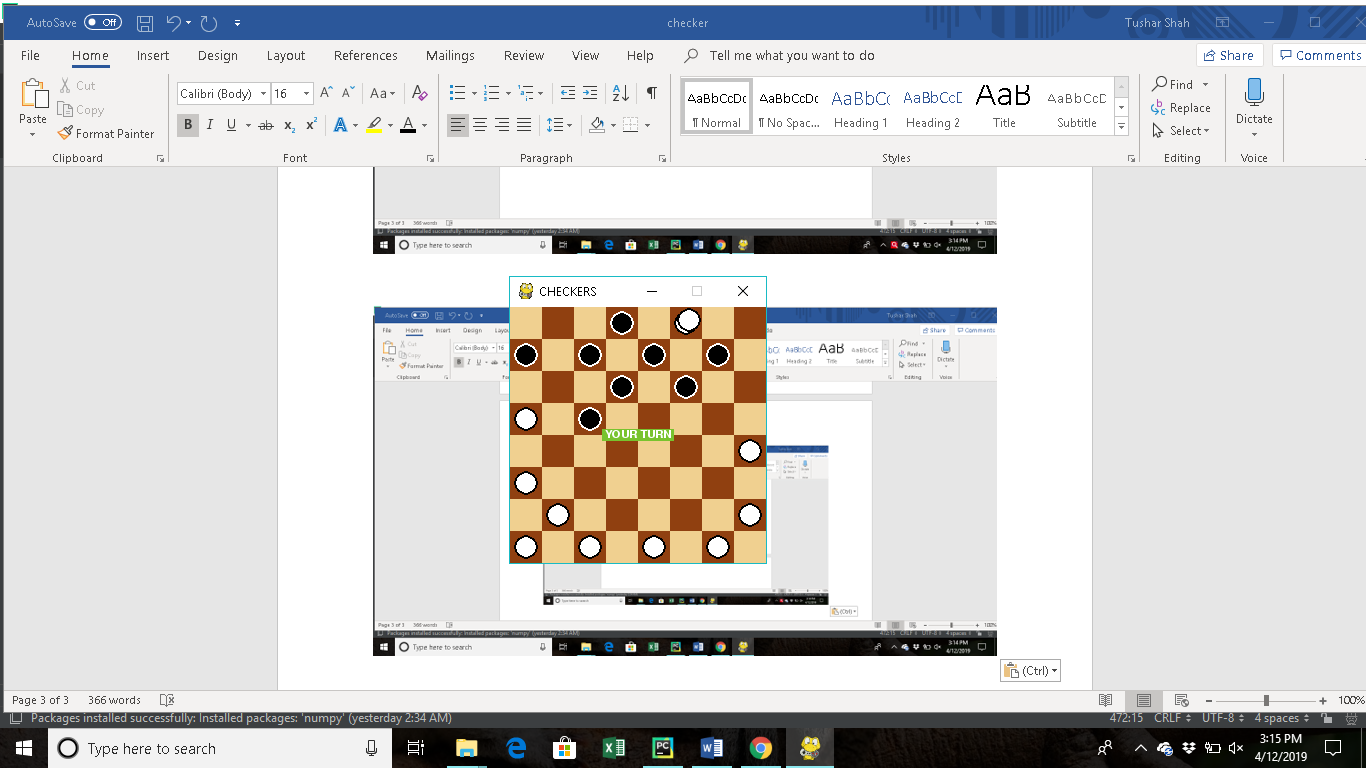
**2.** Eliminating an opponent



**3.Opponent eliminated**



**4.Making king**



**5.Winner winner checker winner**

