```
# Morris_Variant_Game
# CS 6364 - Artificial Intelligence
# Pankhuri Bhatnagar (PXB220012)

<!-- Instructions -->

1. Unzip the folder and import it on your IDE (This project was created on
VSCode).
   The name of the folder is - pxb220012

Programming Language Requirement: Python (preferrably version 3.7+)
Please reach out to me in case of any errors.

2. Add the board input in the board1.txt file.

3. To run any of the 8 files:

[MiniMaxOpening, MiniMaxGame, ABOpening, ABGame, MiniMaxOpeningBlack,
MiniMaxGameBlack, MiniMaxOpeningImproved, MiniMaxGameImproved]

Use the given command:

>> python [filename.py] board1.txt board2.txt [depth]


<!-- Example: -->

>> python ABGame.py board1.txt board2.txt 3
>> python MiniMaxOpeningImproved.py board1.txt board2.txt 2

4. Analyzing Outputs:

<!-- Example 1 -->
## board1.txt input - xWWBxBxxxxxxxxWxBxWxx

>> python MiniMaxOpening.py board1.txt board2.txt 2

    Output:

    Board Position: WWWxxBWxxxWxxxWxBxWxx
    Positions evaluated by Static Estimation: 157
    MINIMAX estimate: 4

    Conclusion: A mill is formed by White player and a Black piece is removed in
the Opening itself.
```

```
>> python MiniMaxOpeningImproved.py board1.txt board2.txt 2

    Output:

    Board Position: WWWBxBxxxxWWxxWxBxWxx
    Positions evaluated by Static Estimation: 157
    MINIMAX estimate: 8

    Conclusion: A mill is formed and a different board position came as output
which has better positioning and higher chances of winning.

## Running the same board position for ABOpening, we get

>> python ABOpening.py board1.txt board2.txt 2

    Output:

    Board Position: WWWxxBWxxxWxxxWxBxWxx
    Positions evaluated by Static Estimation: 68
    MINIMAX estimate: 4

    Conclusion: We get the same output board position as it should be, but the
positions evaluated which was 157 for MiniMaxOpening, came down to just 68 for
ABOpening. This happened because of Alpha-Beta cut and hence, we are not
unnecessarily evaluating all the positions in Alpha-Beta Pruning.



<!-- Example 2 -->
## board1.txt input - xWWBxBxxxxxxxxWxBxWxx

>> python MinimaxGame.py board1.txt board2.txt 2

    Output:

    Board Position: xWWBxBWxxxWxxxWxBxWxx
    Positions evaluated by Static Estimation: 360
    MINIMAX estimate: 2959

>> python MinimaxGameImproved.py board1.txt board2.txt 2

    Output:

    Board Position: WWxBxBxWxxWxxxWxBxWxx
    Positions evaluated by Static Estimation: 360
    MINIMAX estimate: 14964
```

Conclusion: A mill is formed and a different board position came as output which has better positioning and higher chances of winning.

## Running the same board position for ABGame, we get

```
>> python ABGame.py board1.txt board2.txt 2

    Output:

    Board Position: xWWBxBWxxxWxxxWxBxWxx
    Positions evaluated by Static Estimation: 149
    MINIMAX estimate: 2959
```

Conclusion: We get the same output board position as it should be, but the positions evaluated which was 360 for MiniMaxGame, came down to just 149 for ABGame. This happened because of Alpha-Beta cut and hence, we are not unnecessarily evaluating all the positions in Alpha-Beta Pruning.

5. Improved Static Estimation

I have improved the opening static estimation function by taking into consideration 4 more variables for a given board position:
- Number of possible white mills
- Number of possible white moves
- Number of possible black mills
- Number of possible black moves

Number of possible Mills:
For every White piece on the board position, I am checking if there is atleast one more White piece and there is a possiblility of mill for a given board location. If it is present, i increment the count by 1.
I do the same for Black pieces.

Number of possible Moves:
Apart from this, I am also keeping a count of possible white and black moves using generateMoves methods (generateMovesOpening for White and generateMovesBlackOpening for Black)

## Improved Static Estimation - Opening

```
Basically, I am trying to increase the weightage of White piece and removing the
possibilities of Black piece winning by subtracting number of black pieces count,
black moves count and its black mills count from number of white pieces, white
moves count and white mills count. This will enhance the chances of White player
winning by huge margin, which I have displayed in examples given at the end.

My opening static estimation formula:
(num_white + num_white_mills + num_white_moves) - (num_black + num_black_mills +
num_black_moves)

## Improved Static Estimation - Game

I am increasing the weightage of White player's chance to win by considering
number of white moves, white pieces and white mills, which is being multiplied
with 1000. And from this value, I am negating the weighatge of Black pieces by
considering black moves, black pieces and black mills. This will enhance the
chances of White player winning by huge margin, which I have displayed in
examples given at the end.

My game static estimation formula:
1000 * ((num_white + num_white_mills + num_white_moves) - (num_black +
num_black_mills)) - num_black_moves

<!-- Example 3 -->
Input board1.txt:  WxWBxxxxxWWxBxWxBxWxx

>> python MiniMaxOpening.py board3.txt board4.txt 3

    Output:

    Board Position: WxWxxWxxxWWxBxWxBxWxx
    Positions evaluated by Static Estimation: 3504
    MINIMAX estimate: 6

For MiniMaxOpening, it is removing a black piece from position 3

Now with white to play, white will form a mill, and then remove a black piece
instead of removing the black piece which will potentially form a mill in its
next move.

Now, we run the MiniMaxOpeningImproved, and observe the below result.

>> python MiniMaxOpeningImproved.py board3.txt board4.txt 3

    Output:
```

```
    Board Position: WxWBxxxWxWWxBxWxBxWxx
    Positions evaluated by Static Estimation: 3504
    MINIMAX estimate: 21

    As we can see, the white piece formed its mill by adding W at 7th location.
    The MINIMAX estimate changed to 21 (improvement: higher chances of winning),
which was 6 for MiniMaxOpening, while the number of positions evaluated remains
the same.

<!-- Example 4 -->

>> python MiniMaxGame.py board3.txt board4.txt 3

    Output:

    Board Position: xWWBxxxxxWWxBxWxBxWxx
    Positions evaluated by Static Estimation: 3330
    MINIMAX estimate: 2964

>> python MiniMaxGameImproved.py board3.txt board4.txt 3

    Output:

    Board Position: WxxBxxxWxWWxBxWxBxWxx
    Positions evaluated by Static Estimation: 3330
    MINIMAX estimate: 20964

<!-- More Examples -->

# Example 5

>> python MiniMaxOpening.py board3.txt board4.txt 4

Board Position: WxWxxWxxxWWxBxWxBxWxx
Positions evaluated by Static Estimation: 38990
MINIMAX estimate: 5

>> python ABOpening.py board3.txt board4.txt 4

Board Position: WxWxxWxxxWWxBxWxBxWxx
Positions evaluated by Static Estimation: 3588
MINIMAX estimate: 5

Conclusion:
With Alpha-Beta Pruning, the positions evaluated were cut down by 90%.
```

```
From 38990 to 3588 only! A significant reduction.

# Example 6

>> python MiniMaxGame.py board3.txt board4.txt 4

Board Position: xWWBxxxxxWWxBxWxBxWxx
Positions evaluated by Static Estimation: 115205
MINIMAX estimate: 2959


>> python ABGame.py board3.txt board4.txt 4

Board Position: xWWBxxxxxWWxBxWxBxWxx
Positions evaluated by Static Estimation: 6085
MINIMAX estimate: 2959

From 115205 to 6085 only! A significant reduction.
```