1, 2, …, 10

Pieces and Board: This game is played on a 1 by 10 board. The initial position is an empty board.

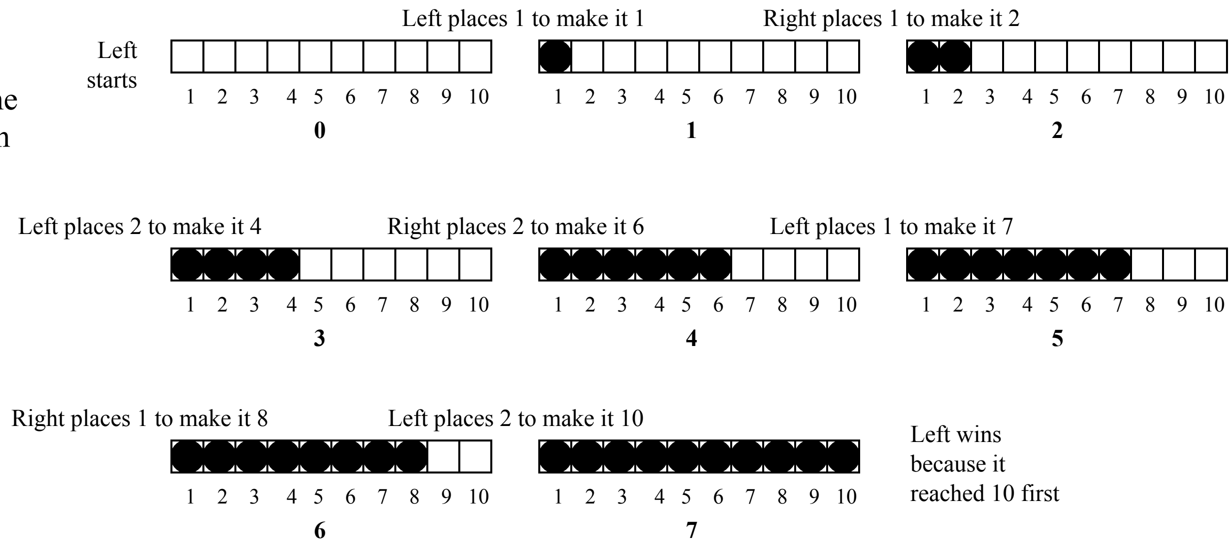
To Move: Players alternate placing either one or two pieces on the leftmost open squares. In this game, we don't distinguish between players' pieces, so we'll call them left and right.

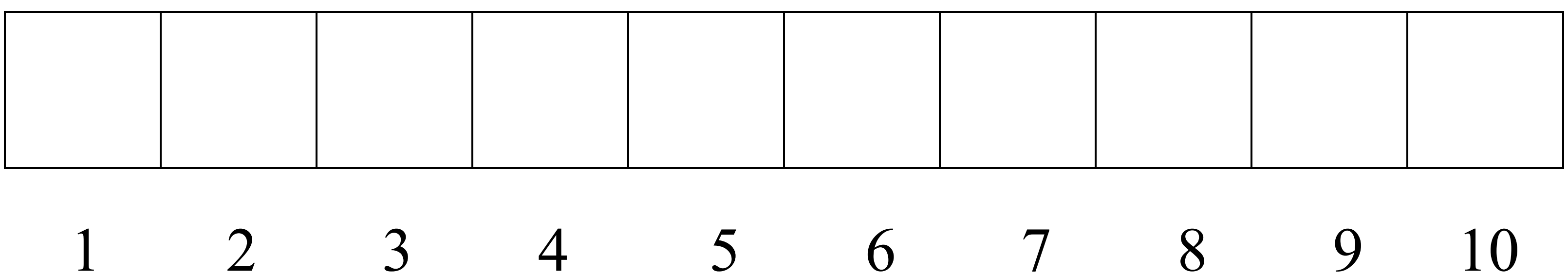
To Win: The first player to place the tenth piece on the board is the winner.

Compulsory Rule Changes:
• Misére Rules: The first player to place the tenth piece on the board is the loser.

Position Representation:
• (*T sum-so-far*)
T stores whose turn it is (L or R), *sum-so-far* stores the sum so far. Example representation for initial position: (L 0)

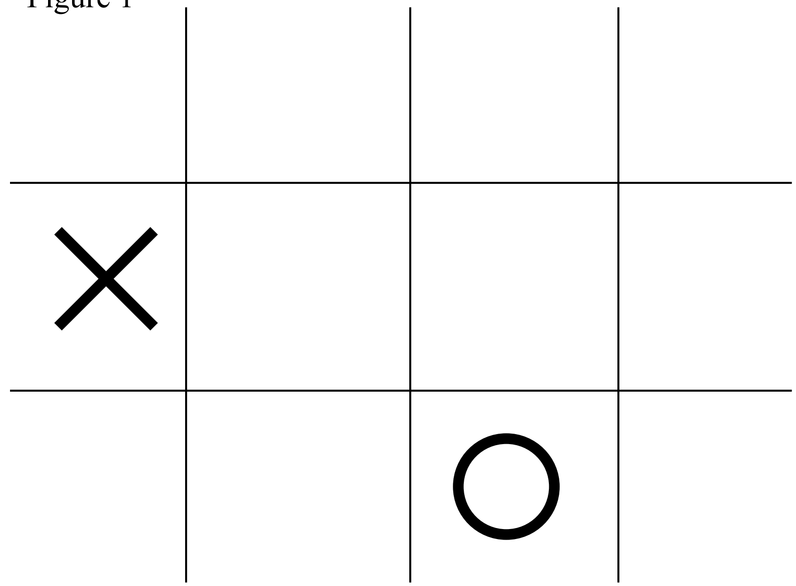
Example game





Tomorrow’s Tic-Tac-Toe

Figure 1



Pieces and Board: This game is played on a rectangular *n* rows by *m* columns board . The default game has 3 rows by 4 columns with the configuration of Figure 1.

To Move: Players alternate placing their pieces (which are usually X’s and O’s) on the board in empty spaces.

To Win: The first player to reach 3-in-a-row (horizontally or vertically) with their pieces wins. If the board is filled and nobody has done this, the game is a tie.

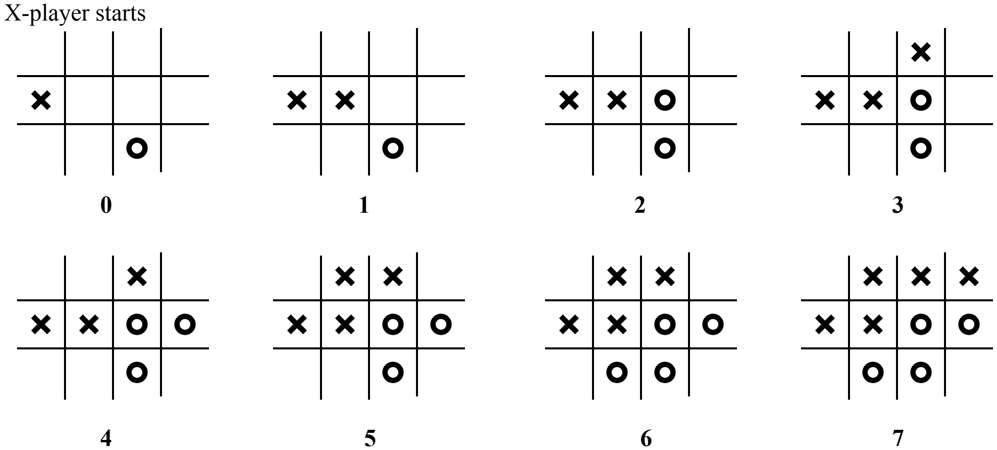
Compulsory Rule Changes:
- Misére Rules: 3-in-a-row loses.
- Allow for diagonal wins.

Position Representation:
- (*T row row row* ...)

T stores whose turn it is (x or ○), Each *row* is in the form *ppp*... where *p* is “x” or “○”, representing the corresponding piece on the board, or “–” if blank. The number of *row*’s in the position indicates the number of rows. In each *row*, the number of *p*’s indicates the number of columns.

Example game:

X-player starts



X-player wins because she has a 3-in-a-row

E.g. representations for initial position (see Figure 1):
`(x ---- x--- --○-)`

sample board mid-game (see board 5 in example game):
`(○ -xx- xx○○ ---○-)`

If you choose to implement this game, you cannot get above a "B" in CS3.

Chopsticks

Pieces and Board: This version of chopsticks is played with *n* hands and *m* fingers. Each hand has initially one finger up. Your implementation must be able to handle an arbitrary number of hands and fingers. The default game has three hands, each having five fingers, with one finger up on each hand and Left starting. (An example game is shown on the right.)

To Move: The example game begins with Figure 1. On the player's turn, she chooses one of her hands to add the number of fingers to one of the opponent's hands.

To Win: When the player adds enough fingers to exceed the max number of fingers that the hand can hold, then that hand is knocked out of the game. When a player has no more hands left, she loses.

Compulsory Rule Changes:

Misére Rules: The player who has all hands knocked out *wins*.

Wrap-Around Rule: In this rule, a hand must *equal* the max number of fingers to be knocked out. Otherwise, the excess number of fingers becomes the new number of fingers on the hand. E.g,. If I have 4 fingers up on a 5-finger hand, and someone adds 1 to that hand, the hand is out. If someone were to add 2 to that hand that's 6, so the hand would wrap around to 6-5=1.

Position Representation:

-(P $Lhand_1$ $Lhand_2$ ... $Rhand_1$ $Rhand_2$ ...)

The P represents the player (L or R). Each hand is a number that shows how many fingers are on that hand. E.g., here's position #7: `(L 0 4 1 0 1 1)`

<table><tr><td>L</td><td>|____</td><td>|____</td><td>|____</td></tr><tr><td>R</td><td>|____</td><td>|____</td><td>|____</td></tr></table>

Figure 1: default initial position

Example Game (L always starts):

1
<table><tr><td>L</td><td>|____</td><td>|____</td><td>|____</td></tr><tr><td>R</td><td>|____</td><td>|____</td><td>|____</td></tr></table>

Left's hand 1 to Right's hand 1

2
<table><tr><td>L</td><td>|____</td><td>|____</td><td>|____</td></tr><tr><td>R</td><td>||___</td><td>|____</td><td>|____</td></tr></table>

Right's hand 1 to Left's hand 2

3
<table><tr><td>L</td><td>|____</td><td>|||__</td><td>|____</td></tr><tr><td>R</td><td>||___</td><td>|____</td><td>|____</td></tr></table>

Left's hand 2 to Right's hand 1

4
<table><tr><td>L</td><td>|____</td><td>|||__</td><td>|____</td></tr><tr><td>R</td><td>|||||</td><td>|____</td><td>|____</td></tr></table>

Right's hand 1 to Left's hand 1

5
<table><tr><td>L</td><td>_____</td><td>|||__</td><td>|____</td></tr><tr><td>R</td><td>|||||</td><td>|____</td><td>|____</td></tr></table>

Left's hand 3 to Right's hand 1

6
<table><tr><td>L</td><td>_____</td><td>|||__</td><td>|____</td></tr><tr><td>R</td><td>_____</td><td>|____</td><td>|____</td></tr></table>

Right's hand 3 to Left's hand 2

7
<table><tr><td>L</td><td>_____</td><td>||||_</td><td>|____</td></tr><tr><td>R</td><td>_____</td><td>|____</td><td>|____</td></tr></table>

Left's hand 3 to Right's hand 2

8
<table><tr><td>L</td><td>_____</td><td>||||_</td><td>|____</td></tr><tr><td>R</td><td>_____</td><td>||___</td><td>|____</td></tr></table>

Right's hand 2 to Left's hand 2

9
<table><tr><td>L</td><td>_____</td><td>_____</td><td>|____</td></tr><tr><td>R</td><td>_____</td><td>||___</td><td>|____</td></tr></table>

Left's hand 3 to Right's hand 2

10
<table><tr><td>L</td><td>_____</td><td>_____</td><td>|____</td></tr><tr><td>R</td><td>_____</td><td>|||__</td><td>|____</td></tr></table>

Right's hand 2 to Left's hand 3

11
<table><tr><td>L</td><td>_____</td><td>_____</td><td>||||_</td></tr><tr><td>R</td><td>_____</td><td>|||__</td><td>|____</td></tr></table>

Left's hand 3 to Right's hand 3

12
<table><tr><td>L</td><td>_____</td><td>_____</td><td>||||_</td></tr><tr><td>R</td><td>_____</td><td>|||__</td><td>|||||</td></tr></table>

Right's hand 3 to Left's hand 3

13
<table><tr><td>L</td><td>_____</td><td>_____</td><td>____</td></tr><tr><td>R</td><td>_____</td><td>|||__</td><td>|||||</td></tr></table>

Left has no hands. Right wins

Abalone

Pieces and Board: Abalone is played on a board with n (≥ 3) rows, m ($26 \geq m \geq 3$) columns and c captures (n+m-3 $\geq c \geq 1$) needed to win. For an arbitrary n,m, black occupies the lower left edges and right the upper-right resulting in n +m-3 pieces each. The NW and SE corners are empty. Your implementation must handle an arbitrary number of rows and columns. The default game has 4 rows by 4 columns with c=2.

To Move: The players, Black and White, take turns moving one their pieces one position in a row, column, or diagonal. This may result in other pieces moving as well.

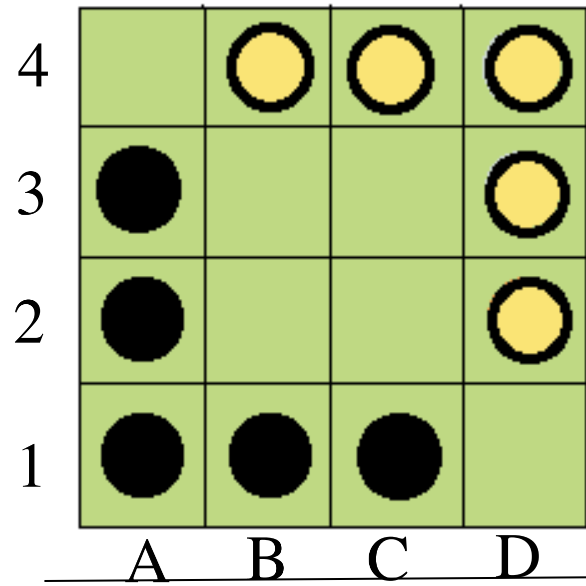
To Push: When one player has numerical superiority in a line or diagonal, she may push the opponent's pieces down the line or diagonal. If the push results in one of the opponent's pieces going off the board, the player captures that piece.

To Win: The player who achieves *c* captures first wins. A player loses if she cannot make a move, and is 'trapped'.

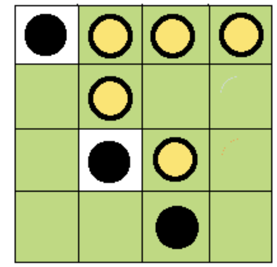
Position Representation: *(T row row row …)* *T* stores whose turn it is (`b` or `w`). Each *row* is in the form *ppp…* where *p* is "`b`" or "`w`", representing the corresponding piece on the board, or "–" if blank. E.g., here's the default board: `(b bbb- b--w b--w -www)`

Compulsory Rule Changes:
Misere Rule: The player who achieves c captures first *loses*. A player *wins* if she cannot make a move, and is 'trapped'.
Freeze-piece: If an unmoved piece could be pushed by opponent on the following turn, the piece is frozen and cannot move.

Default Board (4x4); 2 captures to win. Black to move.

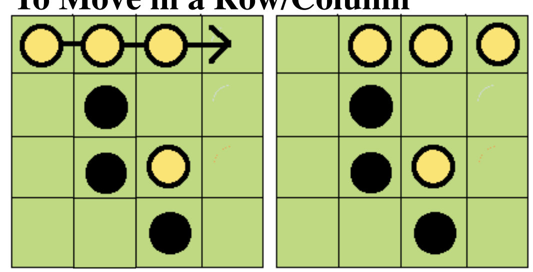


To Freeze Piece:

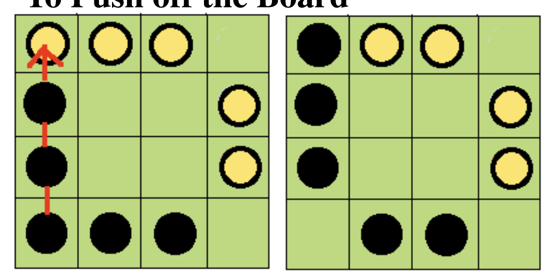


If it is black's turn, the two pieces highlighted in white (b2 and a4) are "frozen," and cannot move because they are in a position to be pushed by the white player.

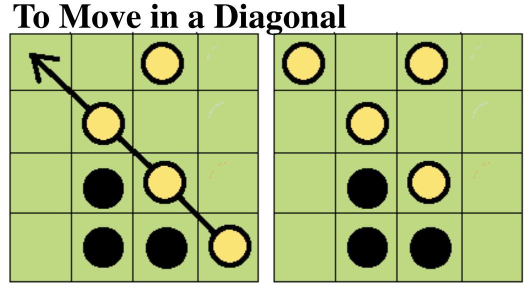
To Move in a Row/Column



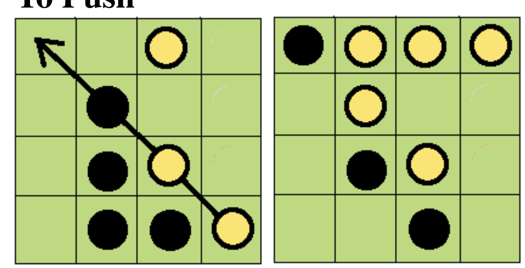
To Push off the Board



To Move in a Diagonal



To Push



Kaboom

Pieces and Board: Kaboom is played on a rectangular *i* rows by *j* columns board. The X pieces belong to X-player, and O pieces to O-player. * symbolizes X-player's bomb, while @ symbolizes O-player's bomb. Your implementation must handle an arbitrary number of rows, columns, and number of pieces needed in a row to win. The example game (Figure 1) has 4 rows by 4 with each player only able to place 1 bomb, and needing 4-in-a-row to win. Pieces have been placed on the board.

To Move: The game begins with a blank board. On one's turn, a player "drops" their piece or bomb into a column from the top. It falls vertically until it reaches another piece, or end of the column. On their turn, a player may choose to detonate a bomb instead of dropping a piece, which removes the bomb and all adjacent pieces to the bomb (i.e. left, right, top, bottom pieces). If another bomb happens to be one of the adjacent pieces, it is also detonated. You must move if you can. I.e., If the board is full but has one of your bombs in it, you must detonate it.

To Win:. n number pieces/bombs in a row from the same player causes a win for that player. Pieces or bombs may connect vertically, horizontally, or diagonally. If both players have n-in-a-row as a result of detonation, it is a draw. If the board becomes full with no n-in-a-rows and no bombs remaining to detonate, it is also a draw.

Compulsory Rule Changes:

• Misére Rules: The player who connects n pieces loses.

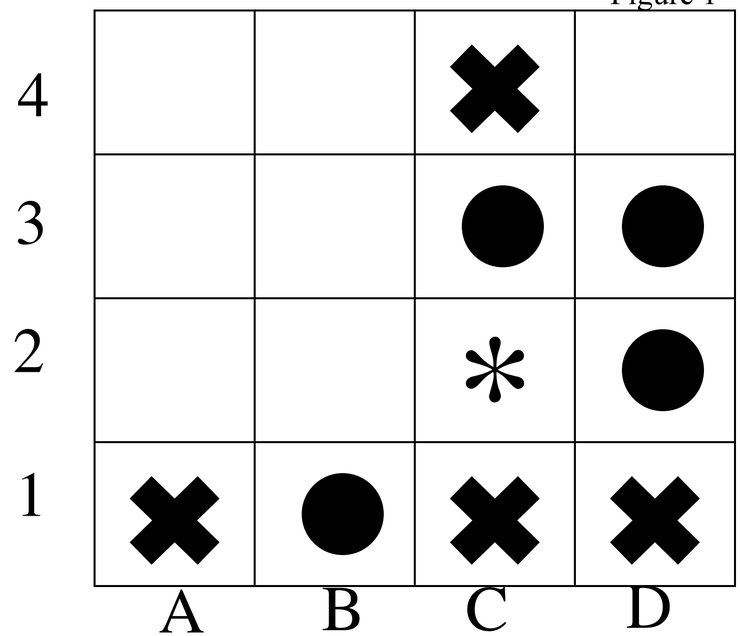
• Super-bomb: bombs remove entire rows and columns

Position Representation:

• (*Player xbombsleft obombsleft row row row …)*

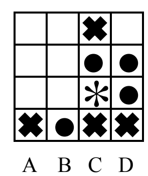
Player stores whose turn it is (`x` or `o`). *xbombsleft* and *obombsleft* store X-player and O-player's number of remaining bombs, respectively. Each *row* is in the form *pppp…* where *p* is "`x`", "`o`", "`*`", or "`@`", representing the corresponding piece or bomb on the board; "`–`" if blank.

Figure 1



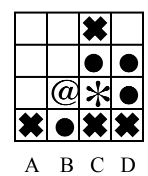
Example game (we came in late, this is position #9, O to play):

…



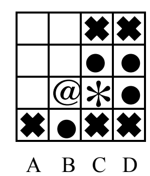
9

O-player, @ to B



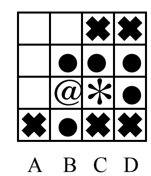
10

X-player, X to D



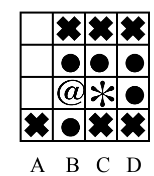
11

O-Player, O to B



12

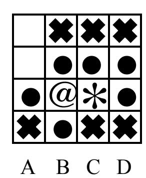
X-Player, X to B



13

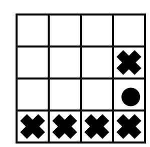
…

O-Player, O to A



14

X-Player, Detonate. Normally that would only take out pieces C1,B2,C2,D2,C3, but since B2 is also a bomb, it explodes, thus also removing B1, A1 and B3. Pieces B4 and C4 fall into B1 and B2…



15

X-Player has four X pieces in a row. X-Player wins.

Representation for position 9: `(o 0 1 xoxx --*o --oo --x-)`

…and position #14: `(x 0 0 xoxx o@*o -ooo -xxx)`

Othello

Pieces and Board: Othello is played on a rectangular board of n (≥ 3) rows and m ($26 \geq m \geq 3$) columns. The default game has 4 rows by 4 columns as shown in Position 0. For an arbitrary n and m, you must center that starting 2x2 cluster for the initial position.

To Move: Each player takes turns adding a piece of their color to the board, such that you 'flip' at least one of the opponents pieces every turn. A piece is flipped if a move results in the piece being sandwiched between an opposing player's pieces. Pieces are flipped in both the orthogonal (left right up down) and diagonal directions (see Figure 2 for an example). If a player cannot flip any pieces, they must pass on their turn.

To Win: The game ends when both players are forced to pass. The player with the most pieces of their own color wins. If the number of pieces is equal, it is a tie.

Compulsory Rule Changes:

• Misére Rules: You win when you have *fewer* points than the opponent when the game ends.

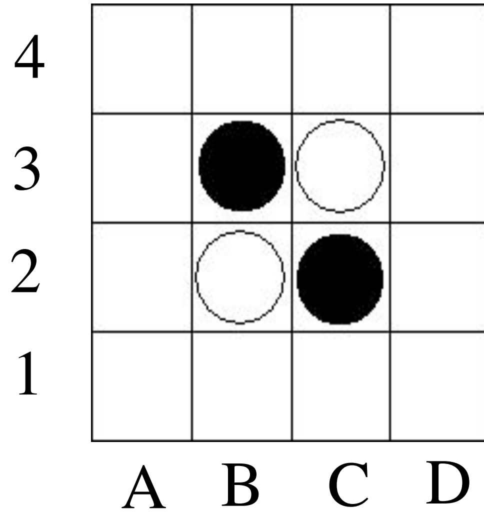
• Place Anywhere: You no longer have to flip a piece every turn, allowing you to place a piece at any location. There are no passes in this mode.

Position Representation:

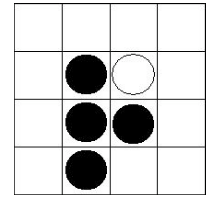
• *(T row row row …)*

T stores whose turn it is (`b` or `w`). Each *row* is in the form *ppp…* where *p* is "`b`" or "`w`", representing the corresponding piece on the board, or "–" if blank. E.g., here's position 1: `(w -b-- -bb- -bw- ----)`

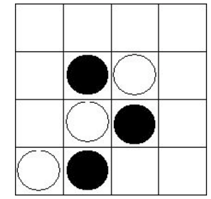
Example game starting with default board (black starts)



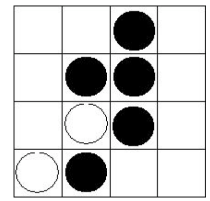
Position 0



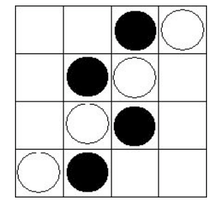
Black at B1, flipping White at B2 1



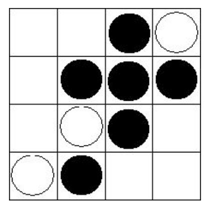
White at A1, flipping Black at B2. 2



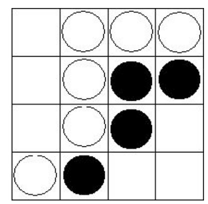
Black at C4, flipping White at C3. 3



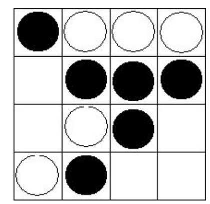
White at D4, flipping Black at C3. 4



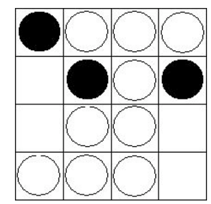
Black at D3, flipping White at C3. 5



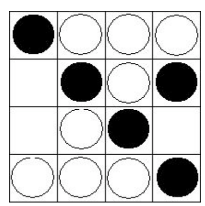
White at B4, flipping Black at B3, C4. 6



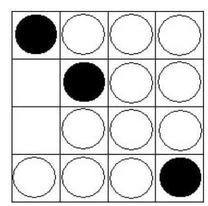
Black at A4, flipping White at B3. 7



White at C1, flipping Black at B1, C2, C3. 8

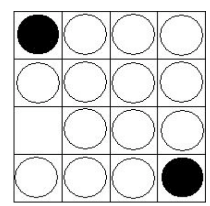


Black at D1, flipping White at C2. 9

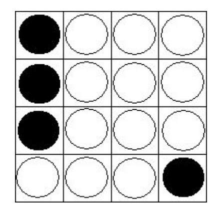


White at D2, flipping Black at C2, D3. 10

Black passes. 11



White at A3, flipping black at B3. 12



Black at A2, flipping White at A3. 13

White wins, 12 to 4.