1, 2, …, 10

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

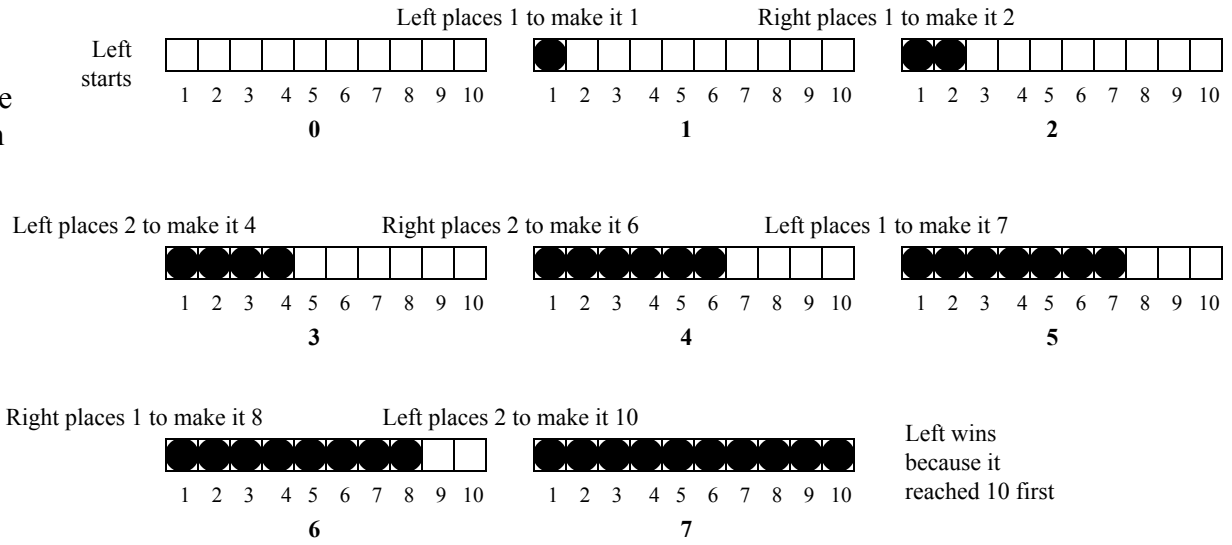
To Move: Players alternate placing either one or two pieces on the leftmost open squares. In this game, we don't distinguish between players' pieces, so we'll call them left and right.

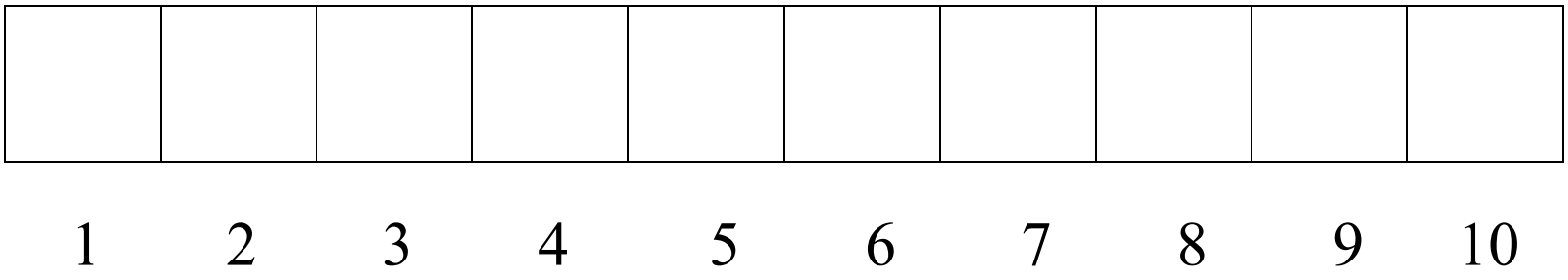
To Win: The first player to place the tenth piece on the board is the winner.

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

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.

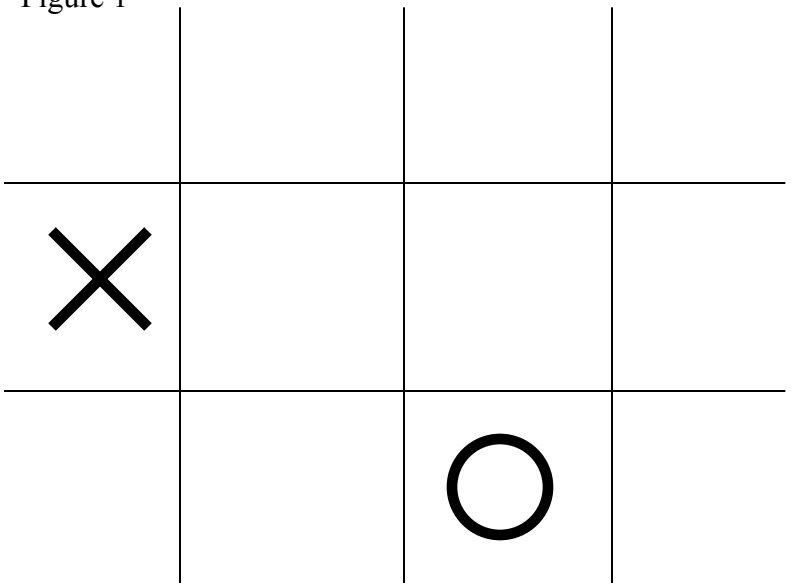
To Win: The first player to reach 3-in-a-row (horizontally or vertically) with their pieces wins. If the board is filled and nobody has done this, the game is a tie.

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

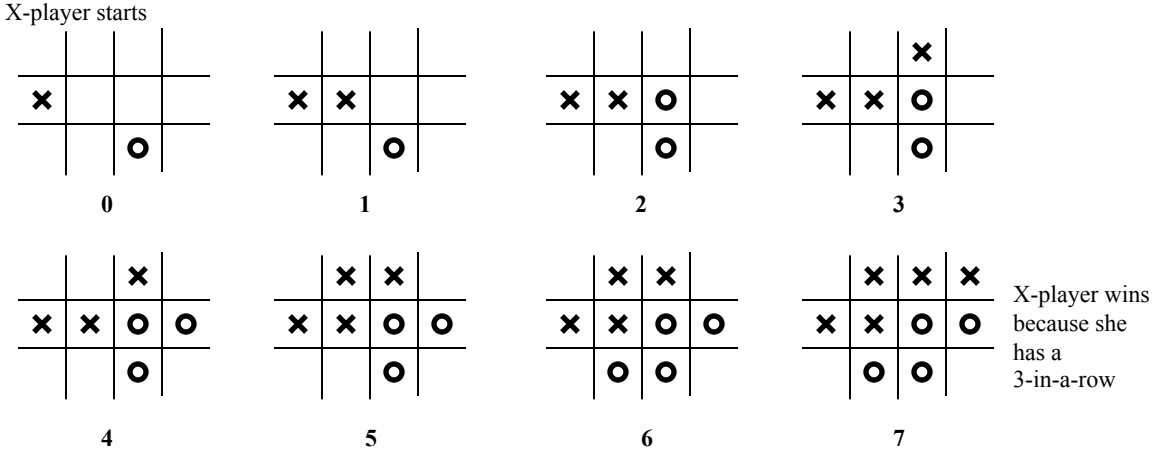
Position Representation:
- (*T row row row ...*)

T stores whose turn it is (`x` or `o`), Each *row* is in the form *ppp...* where *p* is "`x`" or "`o`", 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.

Figure 1



Example game:



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

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

Northcott's Game

Pieces and Board: Northcott's Game is played on an rectangular *n* rows by *m* columns board. The default game has 3 rows of 6 columns with the configuration of Figure 1. In a given row, L may not be placed to the right of R. Your implementation must handle an arbitrary number of rows and columns.

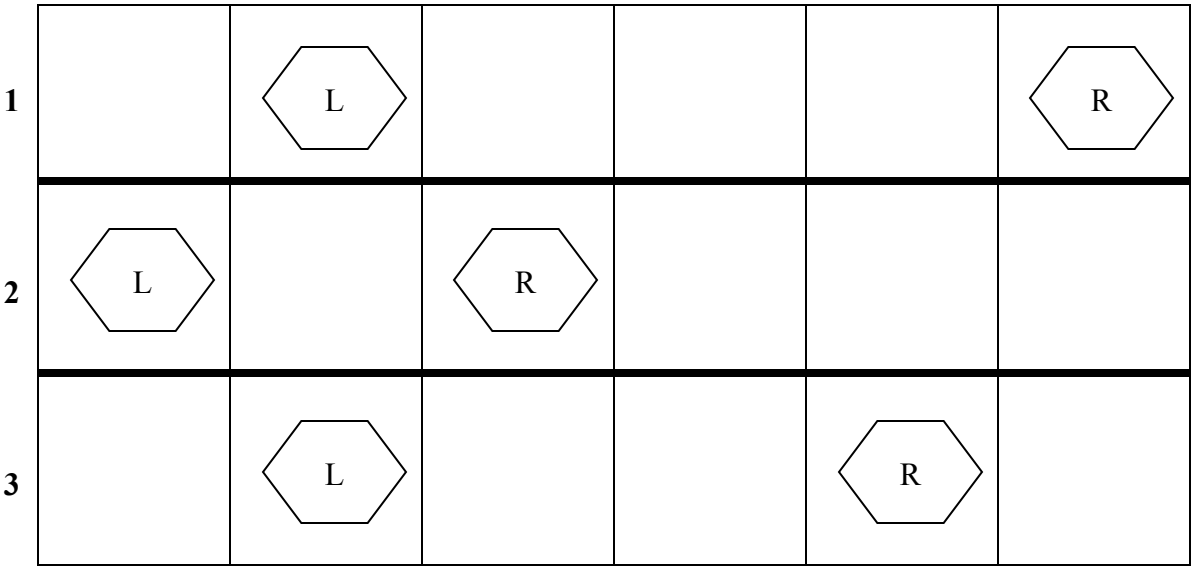
To Move: The players, Left and Right, alternate turns moving as many spaces horizontally forward (towards opponent) without jumping over the opponent. Pieces never leave the row (I.e., they never move vertically).

To Win: The player who has no moves remaining *loses*. (I.e., the last to move wins)

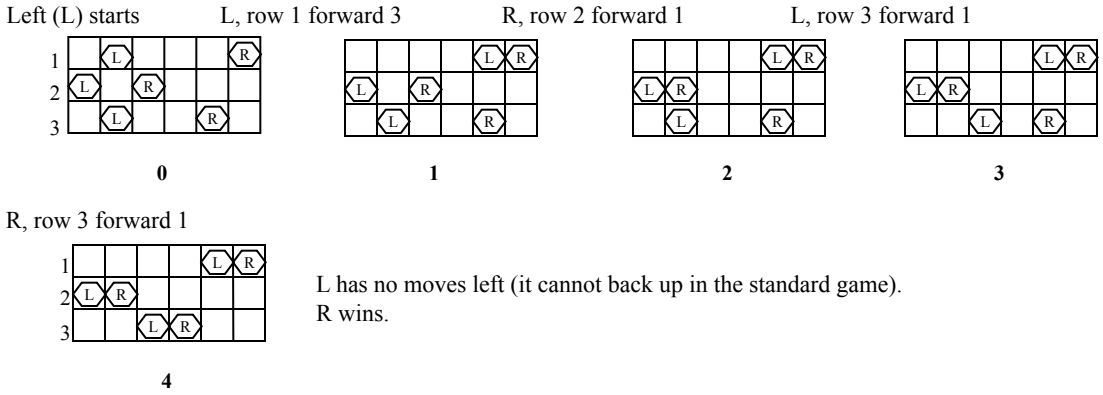
Compulsory Rule Changes:
• Misére Rules: The player who has no moves remaining *wins*. (I.e., the last to move loses)
• A player may move both forward (towards the opponent) and back (away from the opponent).

Position Representation:
• (*T row row row row* …)
T stores whose turn it is (`L` or `R`). Each *row* is a word in the form $\#_1 L \#_2 R \#_3$ where $\#_1$ represents the number of spaces to the left of player L, $\#_2$ the number of spaces between the two players, and $\#_3$ the number of spaces to the right of R. E.g. The row "`- - L - R- - -`" is the word "`2L1R3`". The number of *row*'s in the position indicates the number of rows. In each *row*, the sum of $\#_1$, $\#_2$, $\#_3$, plus 2 (for the pieces) indicates the number of columns.

Figure 1



Example game:



L has no moves left (it cannot back up in the standard game).
R wins.

Representations for initial position (see Figure 1): `(L 1L3R0 0L1R3 1L2R1)`
Representations for position mid-game (see board 1 above): `(R 4L0R0 0L1R3 1L2R1)`

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

Knight's Dance

Pieces and Board: Knight's Dance is played on a rectangular *n* rows by *m* columns board. Each player has a knight and a king. Your implementation must handle an arbitrary number of rows and columns. The default game has 5 rows by 6 columns with the configuration shown here.

To Move: The players, White and Black, take turns moving their knights by "L-shaped" moves as in chess. A player cannot land on her own king. Kings do not move.

To Win: The player who captures (lands on) her opponent's king *or* her opponent's knight wins. A situation where a player does not have any legal moves is a loss for that player. (I.e., if you can't move, you lose!) Ties are no possible, but some games may go on forever.

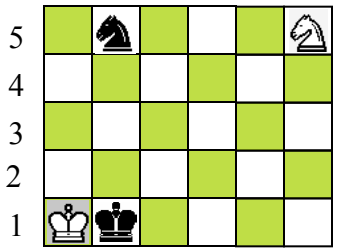
Compulsory Rule Changes:

• Misére Rule: The player who captures (lands on) her opponent's king or her opponent's knight *loses*. If you have no legal moves (e.g., you're blocked in), you win.
• *Jump-block*: A knight is blocked from jumping onto certain squares over pieces it is next to (see illustration).

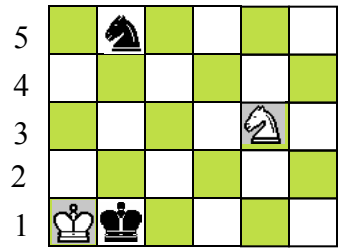
Position Representation:

• (*T #R #C WK BK WN BN*)
T stores whose turn it is (either `B` or `W`). *#R, #C, WK, BK* are fixed throughout a game and represent #rows, #columns, white's king, and black's king respectively. *WN* and *BN* store the slot of the white and black kNight respectively. The slots are represented as in algebraic chess notation (see sample standard game): <column-letter><row-number>, also known as <file><rank>

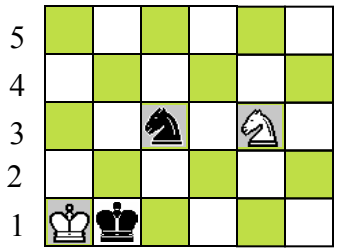
Example game



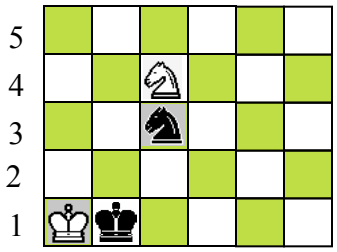
White to move.



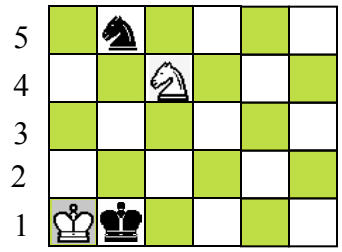
White to e3



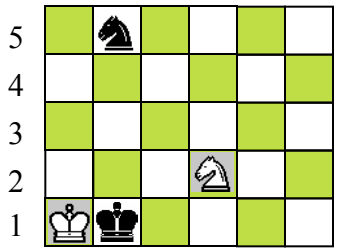
Black to c3



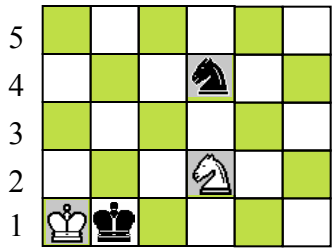
White to c4



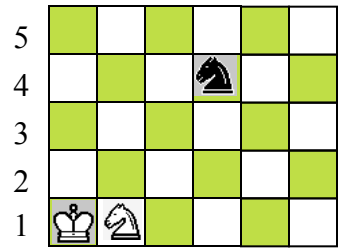
Black to b5



White to d2



Black to d4

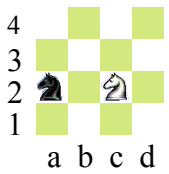


White to b1

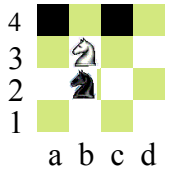
Default initial position:
`(w 5 6 a1 b1 f5 b5)`

White wins by capturing Black's king.

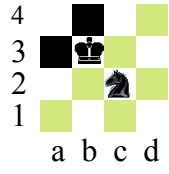
***Jump-block* compulsory rule illustration**



No jump blocks since two pieces not next to each other.



White blocks Black's jumps to a4 and c4 .



Black Knight's jumps to a3 and b4 blocked by own king.

Surround

Pieces and Board: Surround is played on a rectangular board of *n* rows and *m* columns. (see Figure 1) Your implementation must handle an arbitrary number or rows and columns.

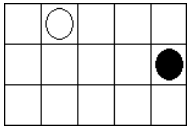


Figure 1
(default board)

To Move: Each player is a worm that is rapidly growing from its head. When it is your turn you must choose the next square for the head to grow onto. You may choose up, down, left, or right, but may not grow diagonally. You must also choose a square not currently occupied by either yourself or your opponent. On your turn, you may not pass or grow off the board.

To Win:. You must cut off your opponent and leave her with no room to grow. You win when your opponent has no room to grow on her turn.

Compulsory Rule Changes:

• **Misére Rules**: You win when *you* have no room to grow on your turn.

• **Worm Holes**: The edges of the board are no longer bounds. Worms may grow into 'worm holes' at the edges allowing the worm to loop around to the opposite side of the board. (see Worm Hole Example)

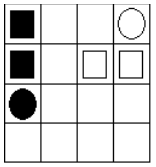
Position Representation:

• (*T row row row row ...*)

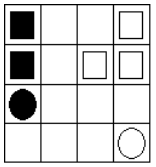
T stores whose turn it is. Each *row* is a word of the form *www…* where *w* is a *slot* on the board. The number of *w*'s is the number of columns and the number of *row's* is the number of rows. Each *w* is either the body or head of the worm consuming the space, or "–" if the space is empty. The head and body of the white worm are represented by '`w`' and '`o`' respectively, while the head and body of the black worm are '`b`' and '`x`' respectively.

Worm Hole Example (on different board):

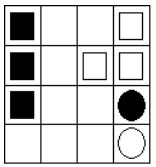
It is white's turn to grow. Its choices are left or worm hole up (it cannot worm hole right because it would grow onto the opponent, nor can it move down because it would grow onto itself). White decides to take the worm hole by moving up.



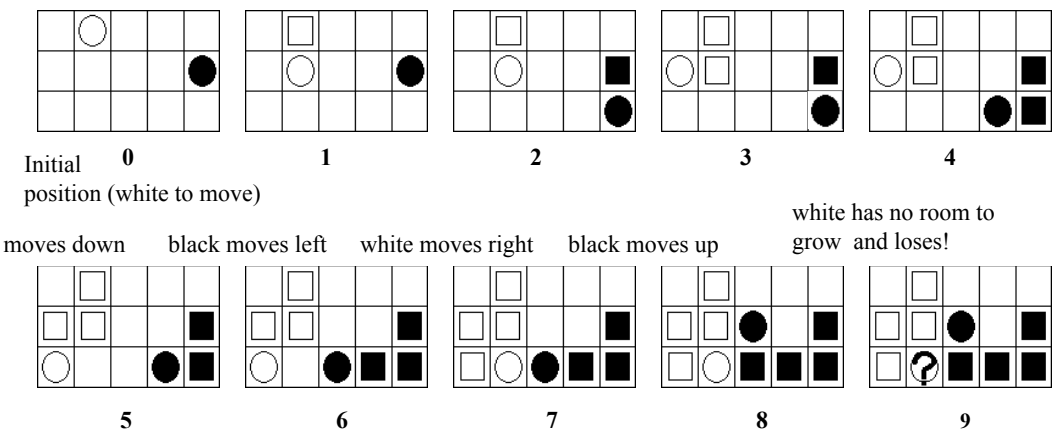
It is black's turn to grow. Its choices are down, right or worm hole left. Black decides to take the worm hole by moving left. It cannot move up because it would grow onto itself.



It is white's turn again and it can choose to grow left or take the worm hole right. It cannot worm hole down because it would grow onto itself, which is an illegal move.



Example game on default board:



E.g., representation for position 0: `(w -w--- ----b -----)`
position 5: `(b -o--- oo--x w--bx)`; position 8: `(w -o--- oob-x owxxx)`

Konane

Pieces and Board: Konane is played on a rectangular *n* rows by *m* columns board. The pieces alternate Black and White, so two pieces of the same color are never adjacent. Your implementation must handle an arbitrary number of rows and spaces. The default game has 3 rows by 4 columns with the configuration of Figure 1.

To Move: The game begins with a board with at least two pieces removed (a black and a white piece). On one's turn, a player uses his/her piece to vertically or horizontally jump over an opponent's piece. Any piece that is jumped is then removed from the board. After a player has just made a move, if the player has more valid moves with the *same* piece, the player may choose to continue and go again (but doesn't have to -- called "passing up a go-again").

To Win:. The player who has no moves remaining loses.

Compulsory Rule Changes:

• Misére Rules: The player who has no moves remaining *wins*.

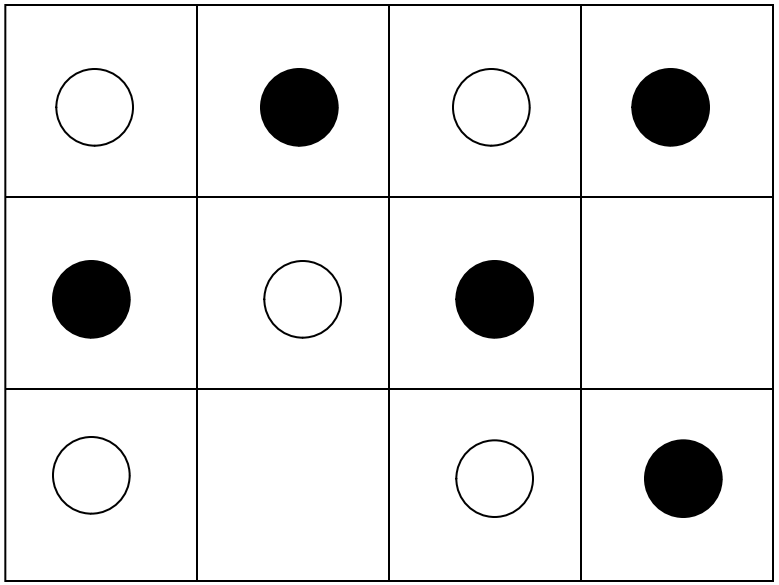
• On a player's go-again turn the player may choose to move a different piece. (I.e., if a player captures a piece on her turn, she may continue to capture pieces if she wishes until there are no pieces left to capture).

Position Representation:

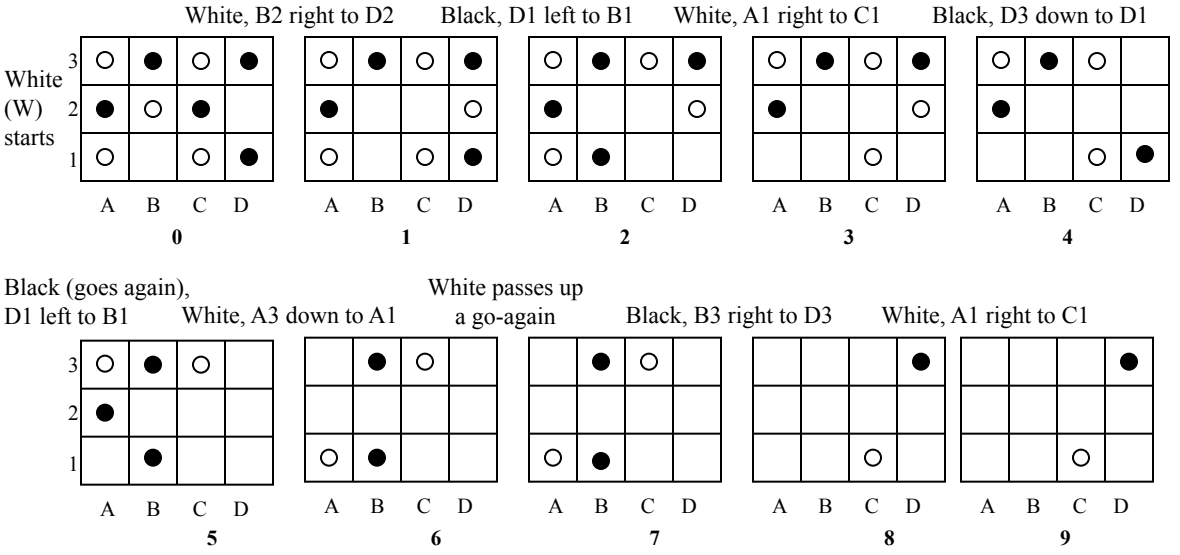
• (*T L row row row row …*)

T stores whose turn it is (`w` or `b`). *L* stores the cell where the player *T* last moved to (used in go-again turns). If the previous move was made by the opponent, *L* should be empty: `""`. Each *row* is in the form *ppp…* where *p* is "`w`" or "`b`", 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.

Figure 1



Example game:



Representation for initial position: `(w "" wbwb bwb- w-wb)`

board #4: `(b d1 wbw- b--- --wb)`