Problem J. Random Chess Game

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

This is an interactive problem.

One a boring Tuesday evening, Jack and Jill decided to play a game of chess. Since Jack is a very mediocre chessplayer, Jill promised to play random moves on her turns. Formally, if on Jill's turn, there are n legal moves, then Jill will choose each move with probability $1/n$. Also, Jill likes black color very much. So, she played all games with black pieces. After losing several games, Jack asked you to write a program which will help him to beat Jill's random strategy.

Chess rules

This section is based on the Wikipedia article about chess.

Chess game pieces are divided into white and black sets. Each set consists of 16 pieces: one king, one queen, two rooks, two bishops, two knights, and eight pawns. The game is played on a square board of eight rows and eight columns. The 64 squares alternate in color and are referred to as light and dark squares. The chessboard is placed with a light square at the right-hand corner nearest to each player. Thus, each queen starts on a square of its own color (the white queen on a light square; the black queen on a dark square).

White moves first, after which players alternate turns, moving one piece per turn (except for castling, when two pieces are moved). A piece is moved to either an unoccupied square or one occupied by an opponent's piece, which is captured and removed from play. With the sole exception of en passant, all pieces capture by moving to the square that the opponent's piece occupies. Moving is compulsory, it is illegal to skip a turn. A player may not make any move that would put or leave the player's own king in check. If the player to move has no legal move, the game is over; the result is either checkmate (a loss for the player with no legal move) if the king is in check, or stalemate (a draw) if the king is not. Each piece has its own way of moving:

- The king moves one square in any direction. The king also has a special move called castling that involves also moving a rook.
- A rook can move any number of squares along a rank or file, but cannot leap over other pieces. Along with the king, a rook is involved during the king's castling move.
- A bishop can move any number of squares diagonally, but cannot leap over other pieces.
- The queen combines the power of a rook and bishop and can move any number of squares along a rank, file, or diagonal, but cannot leap over other pieces.
- A knight moves to any of the closest squares that are not on the same rank, file, or diagonal. (Thus the move forms an L-shape: two squares vertically and one square horizontally, or two squares horizontally and one square vertically.) The knight is the only piece that can leap over other pieces.
- A pawn can move forward to the unoccupied square immediately in front of it on the same file, or on its first move it can advance two squares along the same file, provided both squares are unoccupied. A pawn can capture an opponent's piece on a square diagonally in front of it on an adjacent file, by moving to that square. A pawn has two special moves: the *en passant* capture and *promotion*.

Once in every game, each king can make a special move, known as castling. Castling consists of moving the king two squares along the first rank toward a rook that is on the player's first rank and then placing the rook on the last square that the king just crossed. Castling is permissible if the following conditions are met:

- Neither the king nor the rook has previously moved during the game.
- There are no pieces between the king and the rook.
- The king cannot be in check, nor can the king pass through any square that is under attack by an enemy piece, or move to a square that would result in check. (Note that castling is permitted if the rook is under attack, or if the rook crosses an attacked square.)

When a pawn makes a two-step advance from its starting position and there is an opponent's pawn on a square next to the destination square on an adjacent file, then the opponent's pawn can capture it *en passant* ("in passing"), moving to the square the pawn passed over. This can be done only on the very next turn; otherwise the right to do so is forfeited.

When a pawn advances to the eighth rank, as a part of the move it is promoted and must be exchanged for the player's choice of queen, rook, bishop, or knight of the same color. Usually, the pawn is chosen to be promoted to a queen, but in some cases another piece is chosen; this is called underpromotion. There is no restriction on the piece promoted to, so it is possible to have more pieces of the same type than at the start of the game (for example, two or more queens).

When a king is under immediate attack by one or two of the opponent's pieces, it is said to be in check. A move in response to a check is legal only if it results in a position where the king is no longer in check. This can involve capturing the checking piece; interposing a piece between the checking piece and the king (which is possible only if the attacking piece is a queen, rook, or bishop and there is a square between it and the king); or moving the king to a square where it is not under attack. *Castling* is not a permissible response to a check.

The object of the game is to checkmate the opponent; this occurs when the opponent's king is in check, and there is no legal way to remove it from attack. It is never legal for a player to make a move that puts or leaves the player's own king in check.

There are several ways games can end in a draw:

- *Stalemate*: The player whose turn it is to move has no legal move and is not in check.
- *Threefold repetition*: This most commonly occurs when neither side is able to avoid repeating moves without incurring a disadvantage. In this situation, either player can claim a draw. The three occurrences of the position need not occur on consecutive moves for a claim to be valid. Two positions are considered same or equal if all occupied squares and kind of pieces (not necessarily the same piece) they occupy are the same, the castling rights for both sides did not change, and no *en passant* capture was possible during the first occurrence, even if obviously not played.
- *Fifty-move rule*: If during the previous 50 moves (100 half-moves) no pawn has been moved and no capture has been made, either player can claim a draw. A half-move is a turn by either White or Black.

In this problem we assume that both players claim a draw whenever it is possible.

Standard Algebraic Notation (SAN)

This section is based on the Wikipedia article about algebraic notation in chess.

Each square of the chessboard is identified by a unique coordinate pair: a letter and a number. The vertical columns of squares, called files, are labeled `a` through `h` from White's left (the queenside) to right (the kingside). The horizontal rows of squares, called ranks, are numbered `1` to `8` starting from White's side of the board. Thus each square has a unique identification of file letter followed by rank number.

Each piece type (other than pawns) is identified by an uppercase letter (`K` for king, `Q` for queen, `R` for rook, `B` for bishop, and `N` for knight). Pawns are not identified by uppercase letters, but rather by the absence of one. Distinguishing between pawns is not necessary for recording moves, since only one pawn can move to a given square.

Each move of a piece is indicated by the piece's uppercase letter, plus the coordinate of the destination square. For example, `Qg4` (move a queen to *g4*). For pawn moves, a letter indicating pawn is not used, only the destination square is given. For example, `e4` (move a pawn to *e4*).

When a piece makes a capture, an `x` is inserted immediately before the destination square. For example, `Bxa6` (bishop captures the piece on *a6*). When a pawn makes a capture, the file from which the pawn departed is used to identify the pawn. For example, `fxe7` (pawn on the *f*-file captures the piece on *e7*). *En passant* captures are indicated by specifying the capturing pawn's file of departure, the `x`, and the destination square (not the square of the captured pawn). For example, `exf6` (pawn on the *e*-file captures the pawn on *f5*).

When two (or more) identical pieces can move to the same square, the moving piece is uniquely identified by specifying the piece's letter, followed by (in descending order of preference):

1. the file of departure (if they differ), for example, `Nbc3`,
2. the rank of departure (if the files are the same but the ranks differ),
3. both the file and rank (if neither alone is sufficient to identify the piece, which occurs only in rare cases where one or more pawns have promoted, resulting in a player having three or more identical pieces able to reach the same square).

As above, an `x` can be inserted to indicate a capture. For example, `N1xc3` (white knight on *b1* captures black piece on *c3* when another white knight is located on *b5*).

When a pawn moves to the last rank and promotes, an equals sign and the piece promoted to is indicated at the end of the move notation, for example: `h8=R` (promoting to rook).

Castling is indicated by the special notations `O-O` (for kingside castling) and `O-O-O` (queenside castling). Note that uppercase letter `O` is used.

A move that places the opponent's king in check has the character `+` appended. If check is also a mate then the character `+` is replaced by the character `#`.

Interaction Protocol

Each line of input describes one input command. Each command consists of command type and command argument separated by a colon and a single space. There are three different types of input commands:

`black_move:` *<last-black-move>*
`white_moves:` *<move-list>*
`result:` *<verdict>*

The game starts with a `white_moves` command listing the initially possible moves: *<move-list>* is a space-separated list of legal white moves in some order.

On each turn, your program should choose one legal move for White from the given *move-list* and output it on a single line.

After that, if the game has ended after your program's move, the `result` command is sent. Otherwise, a `black_move` command is sent describing the Black move (recall that it is chosen uniformly at random from all legal moves).

After that, if the game has ended after the Black move, the `result` command is sent. Otherwise, a `white_moves` command is sent again, listing all currently available moves in some order, and then it is your program's turn again.

Your program should terminate after receiving the `result` command. There are six different types of verdict (game termination status):

<verdict>	*result*	*<verdict>*	*result*
`White won by checkmate`	**OK**	`Illegal move`	**PE**
`Game drawn by stalemate`	**WA**	`Game drawn by repetition`	**WA**
`Game drawn by fifty-move rule`	**WA**	`Black won by checkmate`	**WA**

After printing each line, flush the output buffer, or you will get the outcome `Idleness Limit Exceeded`: this can be done by calling, for example, `fflush (stdout)` in C or C++, `System.out.flush ()` in Java, or `sys.stdout.flush ()` in Python.

There are 150 different tests. In each test, the initial state of pseudorandom generator used to generate the moves is fixed in advance. Test 1 corresponds to the example.

Example

standard input	standard output
white_moves: a3 a4 b3 b4 c3 c4 d3 d4 e3 e4 f3 f4 g3 g4 h3 h4 Nh3 Na3 Nc3 ←Nf3	
	e4
black_move: c5 white_moves: a3 a4 b3 b4 c3 c4 d3 d4 f3 f4 g3 g4 h3 h4 e5 Bc4 Nc3 Ba6 Qh5 ←Nf3 Ke2 Na3 Bb5 Qe2 Ne2 Nh3 Bd3 Be2 Qf3 Qg4	
	e5
black_move: Na6 white_moves: a3 a4 b3 b4 c3 c4 d3 d4 f3 f4 g3 g4 h3 h4 e6 Bc4 Nc3 Bxa6 Qh5 ←Nf3 Ke2 Na3 Bb5 Qe2 Ne2 Nh3 Bd3 Be2 Qf3 Qg4	
	Bxa6
black_move: g5 white_moves: a3 a4 b3 b4 c3 c4 d3 d4 f3 f4 g3 g4 h3 h4 e6 Bc4 Nc3 Qh5 Be2 ←Ke2 Na3 Bb5 Kf1 Qe2 Ne2 Nh3 Bd3 Bf1 Bxb7 Nf3 Qf3 Qg4	
	Qg4
black_move: f5 white_moves: a3 a4 b3 b4 c3 c4 d3 d4 f3 f4 g3 h3 h4 e6 exf6 Nc3 Qa4 Qd4 ←Qe2 Ne2 Qf3 Qb4 Qxg5 Na3 Qf4 Qc4 Kf1 Qd1 Qg3 Qh5# Qxf5 Bb5 Qe4 Bd3 Qh4 Bf1 ←Be2 Qh3 Ke2 Nh3 Kd1 Bxb7 Nf3 Bc4	
	exf6
black_move: b5 white_moves: a3 a4 b3 b4 c3 c4 d3 d4 f3 f4 g3 h3 h4 fxe7 f7+ Nc3 Qa4 Qd4 ←Qe2 Ne2 Qf3 Qb4 Qxg5 Na3 Qf4 Qc4 Kf1 Qe6 Qd1 Qg3 Qh5# Qf5 Bxb5 Qe4 Qxd7+ ←Qh4 Bxc8 Qh3 Ke2 Nh3 Kd1 Bb7 Nf3	
	Ne2
black_move: Rb8 white_moves: a3 a4 b3 b4 c3 c4 d3 d4 f3 f4 g3 h3 h4 fxe7 f7+ O-O Nbc3 Nec3 ←Qa4 Qd4 Nd4 Qf3 Qb4 Rg1 Qxg5 Na3 Qf4 Qc4 Kf1 Ng1 Qe6 Qg3 Qh5# Qf5 Bxb5 Ng3 ←Qe4 Rf1 Qxd7+ Qh4 Bxc8 Qh3 Nf4 Kd1 Bb7	
	Qh5#
result: White won by checkmate	

The "←" characters and blank lines were inserted into example to enhance readability. Real input doesn't contain such characters, and there are no empty lines in the input.

Explanation

In the example above, you could see an *en passant* capture (`5. exf6`). Also, on white move 7, there are examples of *disambiguating moves* (`Nbc3` and `Nec3`) and *castling* move (`O-O`).