KALAH GAME

Kalah is played on a board with two rows of six holes facing each other. Each player owns a row of six holes, plus a kalah to the right of the holes. In the initial state, there are six stones in each hole and the two kalahs are empty.

A player begins his move by picking up the stones of one of his holes. Proceeding counterclockwise around the board, he puts one of the picked-up stones in each hole and in his own kalah, skipping the opponent's kalah, until no stones remain to be distributed. There are three possible outcomes. If the last stone lands on the kalah, the player has another move. If the last stone lands on an empty hole owned by the player, and the opponent's hole directly across the board contains at least one stone, the player takes all the stones in the opponent hole plus his last landed stone and puts them all in his kalah. Otherwise, the player's turn ends, and his opponent moves.

If all of the holes of a player become empty (even if it is not his turn to play), the stones remaining in the holes of the opponent are put in the opponent's kalah and the game ends. The winner of the game is the player that has more stones in his kalah.

The board is represented as a 4 arguments structure

***board (Holes, Kalah, OppHoles, OppKalah),* where**

***Holes* is a list of the numbers of stones in your six holes**

***Kalah* is the number of stones in your kalah**

***OppHoles* is a list of the numbers of stones in the opponent's holes**

***OppKalah* is the number of stones in the opponent's kalah**

A move consists of choosing a hole and distributing the stones. A move is specified as a list of integers with values between 1 and 6 inclusive, where the numbers refer to the holes. Hole 1 is farthest from the player's kalah, while hole 6 is closest. A list is necessary rather than a single integer, because a move may continue.

The predicate for making a move is

***distribute_stones (Stones, N, Board, Board1),* which computes the relation that *Board1* is obtained from *Board* by distributing the number of stones in *Stones*, starting from hole number N.**

There are two stages to the distribution, putting the stones in the player's holes and putting the stones in the opponent's holes. The simpler case is distributing the stones in the opponent's holes. The distribution of stones continues recursively if there is an excess of stones. A check is made to see if the player's board has become empty during the course of the move, and, if so, the opponent's stones are added to his kalah.

Distributing the player's stones must take into account two possibilities, distributing from any particular hole, and continuing the distribution for a large number of stones.

The evaluation function is the difference between the number of stones in the two kalahs

value (board (H, K, Y, L) , Value) :- Value is K - L.