**Nannon**

**Lina Ghanim**

Representation:

\*DIE\* : max number on dice

\*pos\* : max number of positions on the board

**\***ch**\* :**  number of checkers for **both** players.

\*start\* : starting position on the board.

(0 1 2 5 6 7) : each list like this is a representation of positions on the board. Each number represents the position of a checker. First 3 numbers are the positions of player 1’s checkers. Last 3 numbers are the positions of player 2’s checkers.

Positions: 0 and 7 are positions off the board, 0 is on player 1’s side and 7 is on player 2’s side. (7 = \*pos\*+1)

The winner: player 1 wins if first 3 positions are all 7s and player 2 wins if last 3 positions are all 0s.

Point of view: the game is always played in terms of the point of view of the first player:

* Winning means that all its positions have to be 7.
* A move is played from the first 3 positions only.

**Methods**:

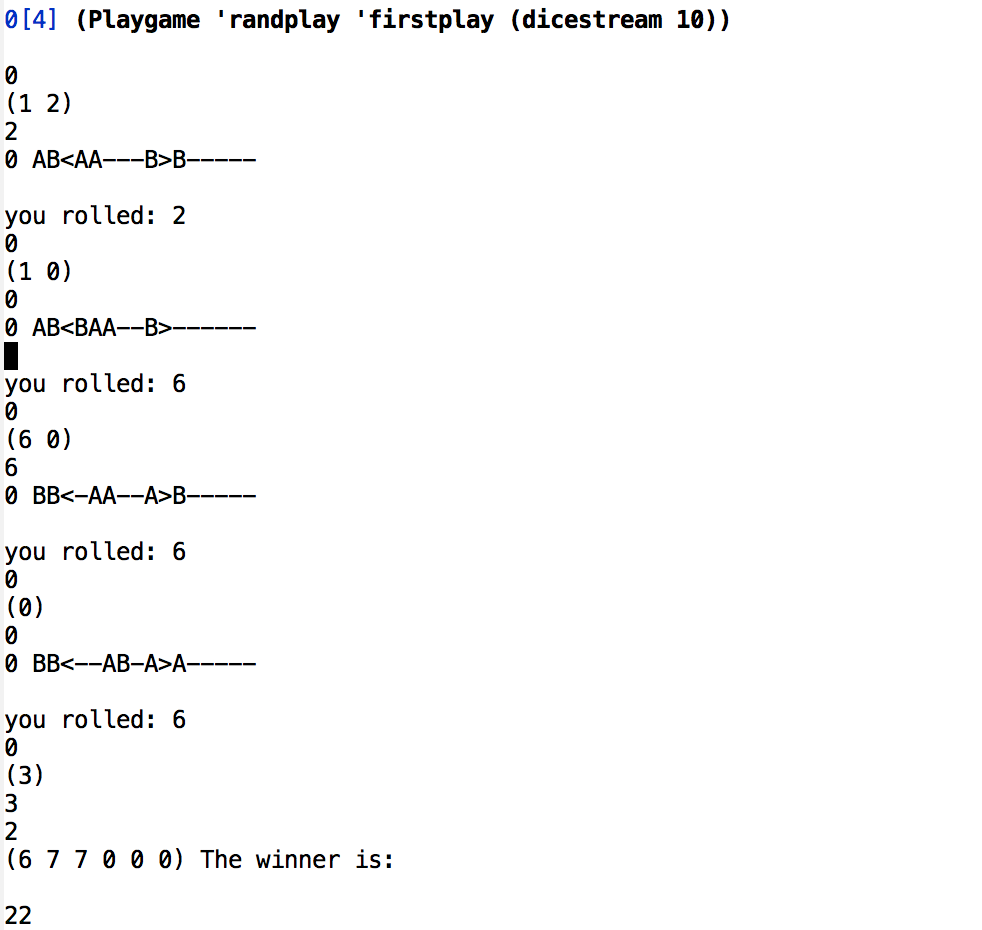
* Swapplayer: reverses the positions for player 1 and 2 and changes their point of view.

For example: for both player 1 and 2 their starting positions list is the same: (0 1 2 5 6 7)

* Playgame: plays between 2 strategies and reports the winner.

Example: (Playgame 'randplay 'firstplay (dicestream 100))

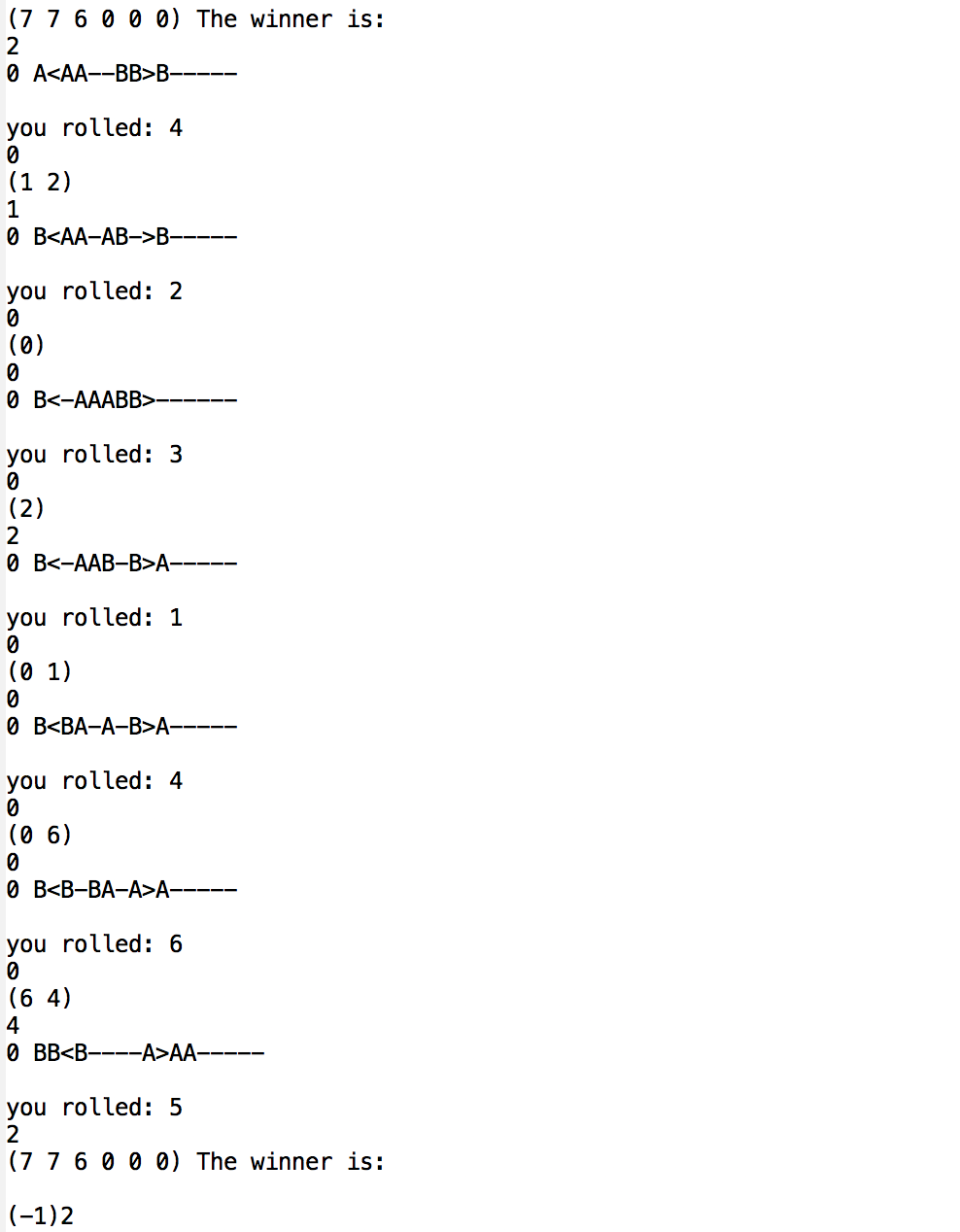
Output: The winner is: 2



* Playtourn: plays n games between 2 strategies and passes the same dice stream between them for each game. Outputs a number between -1 to 1 stating how often strategy 1 wins.

Example: (Playtourn 'randplay ‘firstplay 10)

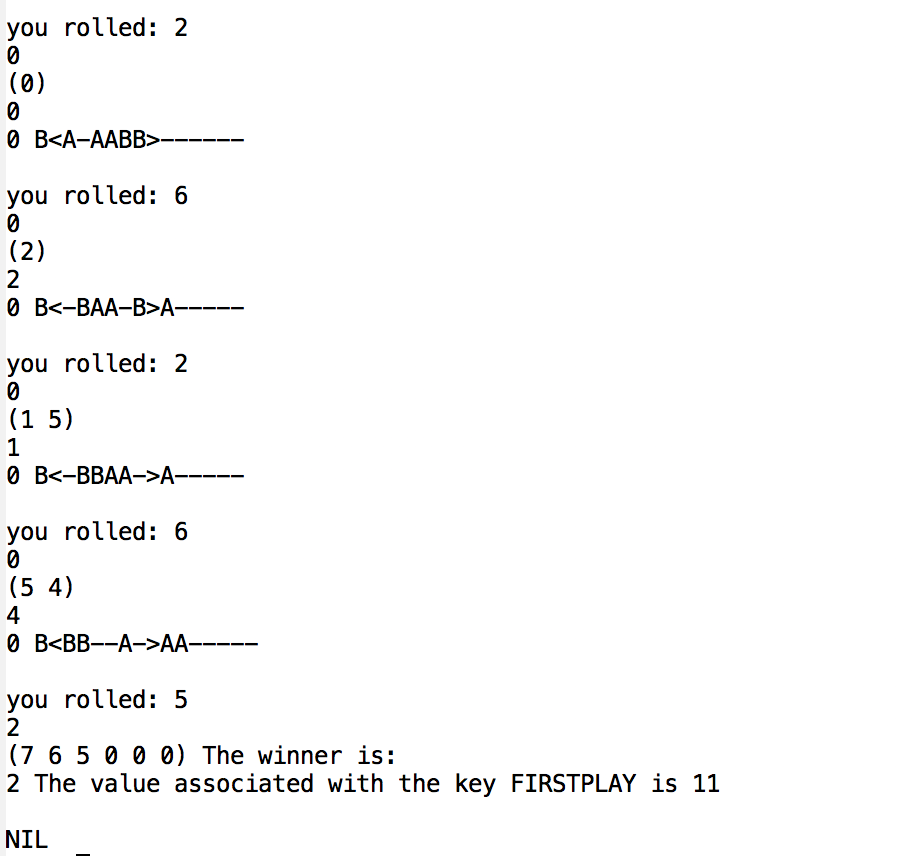
Output: -1 🡨 strategy 2 always wins



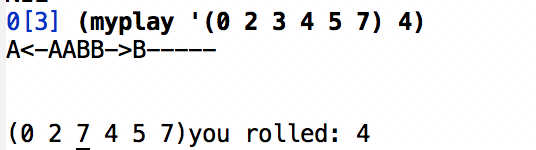
* Roundrobin: a special ordering of different 2 players competing each time in a game (n games). It stores the answers in a hashtable of each player and how many times they won.

Example: (roundrobin (list 'randplay 'firstplay) 10)

Hash table \*scores\* : The value associated with the key FIRSTPLAY is 11



* Mystrategy: my strategy in playing the game is first checking if there is a move that would get me to the other side. If yes, I take it. If not, I do the last play strategy which chooses the checker closest to the end.

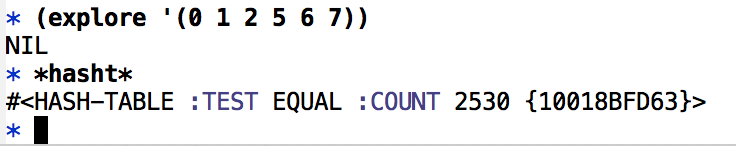


* Explore: this method explores every legal move. I start with the starting position, check all the legal moves from that position, take the move and continue exploring from there. It also explores after swapping each time.

Note: positions of checkers don’t matter as long as they are for the same player. For example: (1 2 3) and (2 1 3) are the same thing since checkers are not distinguishable. That is why I sort the list before adding it to the hash table.

Input: (explore '(0 1 2 5 6 7))

Output: hash table \*hasht\* with count: 2530



Note: whenever you test a method that uses a hashtable, you need to initialize the hashtable again so it deletes the past values in it.