Basic Concepts

The heart of this solver is the function win_move. Make sure you understand how it works as it is very important to the function of hex_simple. On each call, a "mustplay" region is initialized to a set of all the empty cells on the board. While the mustplay region is non-empty, a move is chosen from it. That move is made and then if the board is now connected for the current player, the board is set back to its previous state and a set containing that move is returned. This set is a winset for the current player. Otherwise, a recursive call is made to see whether the opposing player has a winning move. If there is no winning move for the opposing player, the call to win_move for the opposing player will have returned a winset for the current player. This winset combined with the current move is a new winset for the current player, so it is returned (after the current move is undone). In the case that there was a winning move for the opposing player, the call to win_move for the opposing player will have returned a winset for the opposing player. This winset is unioned with the set of opponent win threats (ovc in the code) and the mustplay is intersected with this winset also, because the current player must play inside this region to block the opposing player from winning. Then the current move is undone and the whole process repeats until the mustplay is empty (meaning there is no winning move for the current player) or a winning move for the current player is found.

Other important functions include get_connections and its helper function connected_cells. The function get_connections finds for each cell on the board that is empty or occupied by the current player which other cells it is connected to. Let the current player be denoted by P1 and the opposing player be denoted by P2. Two cells c1 and c2 are connected for P1 if neither of them are occupied by P2 and either they are adjacent or there is a chain of cells occupied by P1 between them. These functions are used to tell if a player has won. As it is now, this implementation can handle 2x2 and 3x3 hex.

Further reading:

https://webdocs.cs.ualberta.ca/ hayward/papers/puzzlingHexPrimer.pdf