Implementation is as in *advanced_heuristic* function in *checkers.py*.

My own checker game heuristic consists of the following situations:

1. If in a state red already wins, it should always be prioritized.
    a. Say in a state there is only 1 red piece left, it has utility of 1; while some other states of higher utility might not necessarily win.
    b. We should place a very high value of winning state to be 100.
2. When pieces are on in the boarders, they are stable and cannot be captured. If home rows are covered, the other player cannot become king
    a. We should place a higher value of those states, add 1 to each piece at borders, including each piece at its home rows
3. It is preferable to stick together when you advance.
    a. Add 1 to each piece that are protected by two or more checkers
4. The number of moves you can move, the more flexibility you have control over the checkers, given the current configuration.
    a. If the number of moves you can make is more than your opponent, add 1.