The details and the performance of the three heuristics implemented are as follows:

1. custom_sore(game, player) : This heuristic computes the reduction in the opponent player's available moves for each of the legal moves the given player makes when the user specified depth is reached.
   The various steps in the implementation of this function are as follows

   (a) Record the available moves for the given player and the opponent player
   (b) Loop over the moves of the given player:
      (i) Apply the given player's move and compute the new game state
      (ii) Compute the total number of legal moves the opponent player has left due to the given players move
      (iii) Compute the difference between the opponent players available moves before and after the given player's move and append it to "*opp_moves_redn*" array
   (c) Return the maximum value from the "*opp_moves_redn"* array

   This heuristic given an overall 78% of wins for a total of 280 games played using this heuristic

2. custom_score_2(game, player): This function computes the Euclidean distance between the opponent player's current position and all the legal available moves of the given player and returns the max value. This heuristic given an overall 76% of wins for a total of 280 games played using this heuristic.

   ```
   for move in own_moves:
      dist.append(math.sqrt((move[0]-opp_moves[0])**2 + (move[1]-opp_moves[1])**2))
   return max(dist)
   ```

3. custom_score_3(game, player): This function returns the difference between the weighted number of given player's available moves and the opponent player's available moves.

   val = own_moves/2 – opp_moves

   This heuristic given an overall 71% of wins for a total of 280 games played using this heuristic.