DATA STRUCTURE & ALGORITHM

GROUP MEMBER

* JAYPATHY
* SHREESHANTH
* THARANI
* KHUSHAL
* KRISH

QUESTION: -

Develop the chess game to store defence moves of player using suitable data structures?

To store the defence moves of a player in a chess game using data structures, you can use a hash table where each key represents a unique chess board position and its associated value is a list of defence moves. The hash table can be implemented using any hash table data structure such as dictionaries in Python or unordered map in C++.

Each time a player makes a move, the current board position can be hashed and used as a key to retrieve the list of defence moves. If the position is not present in the hash table, a new entry can be created with the position as the key and an empty list as the value. If the opponent makes a threatening move, the current player can add the defence move to the list associated with the current board position.

This way, the defence moves for each unique board position can be easily stored and retrieved, reducing the need for searching through all possible moves to find a suitable defence.

ALGORITHM: -

**Here is an algorithm to develop a chess game that stores defense moves of a player using data structures:**

**Initialize an empty hash table defense moves**

**While the game is in progress:**

**a. Hash the current board position and use it as a key to retrieve the list of defense moves from the defense moves hash table**

**b. If the current position is not present in the hash table, add a new entry to the hash table with the current position as the key and an empty list as the value**

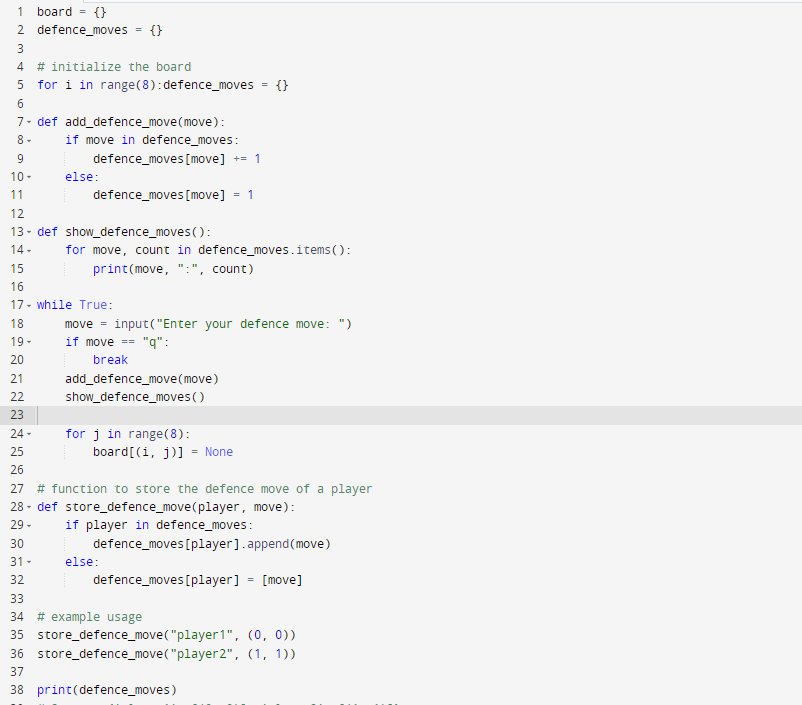
**c. If the opponent makes a threatening move, add the defense move to the list associated with the current board position in the defense moves hash table**

**d. The player can use the defense moves stored in the hash table to make a move**

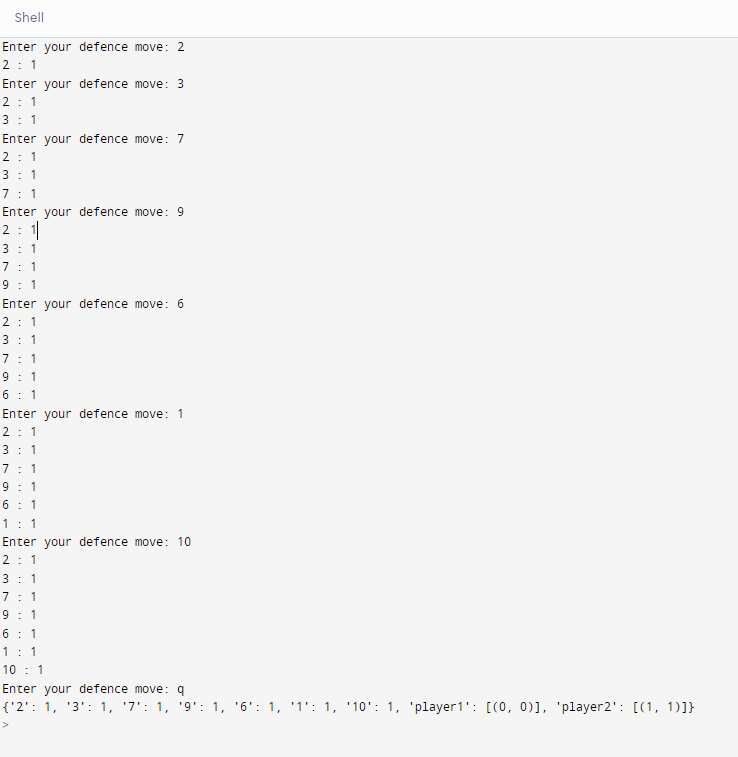
**e. Repeat steps a-d for each player until the game is over**

**Note: The hash function used to hash the board positions should produce unique hashes for each unique position to avoid collisions in the hash table.**

**CODE :-**

****

**OUTPUT :-**

****