**Experiment No: 05**

**Aim:** Write a program for adversarial search (any game)

**Software Used: C/C++/Java/Python**

**Theory**

Adversarial search is a game-playing technique where the agents are surrounded by a competitive environment. A conflicting goal is given to the agents (multiagent). These agents compete with one another and try to defeat one another in order to win the game.

Minimax a decision rule used in decision theory, game theory, statistics and philosophy for minimizing the possible loss for a worst case (maximum loss) scenario. Originally formulated for two-player zero-sum game theory, covering both the cases where players take alternate moves and those where they make simultaneous moves, it has also been extended to more complex games and to general decision making in the presence of uncertainty.

The minimax function returns a heuristic value for leaf nodes (terminal nodes and nodes at the maximum search depth). Non leaf nodes inherit their value, best Value, from a descendant leaf node. The heuristic value is a score measuring the favour-ability of the node for the maximizing player. Hence nodes resulting in a favourable outcome (such as a win) for the maximizing player have higher scores than nodes more favourable for the minimizing player. For non-terminal leaf nodes at the maximum search depth, an evaluation function estimates a heuristic value for the node. The quality of this estimate and the search depth determine the quality and accuracy of the final minimax result.

Minimax treats the two players (the maximizing player and the minimizing player) separately in its code. Based on the observation that.

.



**Formalization of the problem:**

**A game can be defined as a type of search in AI which can be formalized of the following elements:**

* **Initial state:** It specifies how the game is set up at the start.
* **Player(s):** It specifies which player has moved in the state space.
* **Action(s):** It returns the set of legal moves in state space.
* **Result(s, a):** It is the transition model, which specifies the result of moves in the state space.
* **Terminal-Test(s):** Terminal test is true if the game is over, else it is false at any case. The state where the game ends is called terminal states.
* **Utility(s, p):** A utility function gives the final numeric value for a game that ends in terminal states s for player p. It is also called payoff function. For Chess, the outcomes are a win, loss, or draw and its payoff values are +1, 0, ½. And for tic-tac-toe, utility values are +1, -1, and 0.

**Conclusion:**