**Algorithm of Connect 4**

Main Method:

* Declare an integer 2D array for grid
* Declare integer variables(input1, input2) for players’ input (which column to drop their discs, etc)
* Assign the grid with 0s which means the board is clean(no input is received).
* Player 1’s moves will be represented as 1s, and player 2(AI)’s moves will be represented as 2s.
* Players will be allowed to enter which column they want to drop their discs in and the program will change the grid values accordingly in a separate method.

public static int GameResult(int[][] grid)

* After each change, this method is called to check win condition for each player
* The method will check the entire grid for winning conditions for both players
* It checks up, left, right, diagonal up-left, diagonal up-right directions for winning conditions.
* end of method

public static int Minimax(int[][] grid,int turn, int depth)

* After each turn, this method will be called to look ahead number of moves to determine what ai ‘s next move is.
* Create variables called max which stores +infinity, min which stores -infinity, move which stores the best move for ai.
* if ai’s turn:
  + ai is the max player in this case, which will always try to achieve maximum score in each gamestate
  + Recursive call which turn will be changed to player and depth will decrease by 1(depth is the number of moves to look ahead);
  + Create a variable called currentScore which stores the new score by the recursive call.
  + if currentScore > max
    - max = currentScore
    - move = column used in the gamestate.
  + return max
* if player’s turn
  + player is the min player in this case, which will always try to achieve minimum score in each gamestate.
  + Recursive call which turn will be changed to ai and depth will decrease by 1(depth is the number of moves to look ahead);
  + Create a variable called currentScore which stores the new score by the recursive call.
  + if currentScore < min
    - min = currentScore
    - move = column used in the gamestate.(to prevent player from winning);
  + return min
* if depth == 0
  + call Evaluate(grid);
* end of method

public static void Evaluate(grid)

* The method will evaluate the grid for each gamestate.
* For every gamestate, check blanks(positions that has no value, means no player has their disc on the position) on left and right and keep track of it
* Calculate scores for each gamestate, choose the one with the most score.
* if the move can cause a winning condition, take it
* if the move can prevent the opponent from winning, take it
* if the move is neutral(does not satisfy any of the condition above),Choose the base move with the least blanks on either left or right

public static void Print(int[][] grid)

* Print the grid (the initial grid and updated grid after each change)