La función de jugador debe tomar un estado del tablero como entrada y devolver el turno de qué jugador es.

(ya sea X u O).

o En el estado inicial del juego, X obtiene el primer movimiento. Posteriormente, el jugador alterna con cada movimiento adicional.

o Cualquier valor devuelto es aceptable si se proporciona una placa de terminales como entrada (es decir, el juego ya ha terminado).

 La función de acciones debe devolver un conjunto de todas las acciones posibles que se pueden tomar en un tablero dado.

o Cada acción debe representarse como una tupla (i, j) donde i corresponde a la fila del movimiento (0, 1 o 2) y j corresponde a qué celda de la fila corresponde al movimiento (también 0, 1, o 2).

o Los movimientos posibles son las casillas del tablero que aún no tienen una X o una O en ellas.

o Cualquier valor devuelto es aceptable si se proporciona una placa de terminales como entrada.

 La función de resultado toma una placa y una acción como entrada, y debería devolver un nuevo estado de placa, sin modificar la placa original.

o Si la acción no es una acción válida para la junta, su programa debe generar una excepción.

o El estado del tablero devuelto debe ser el tablero que resultaría de tomar el tablero de entrada original y dejar que el jugador al que le toca hacer su movimiento en la celda indicada por la acción de entrada.

o Es importante destacar que la placa original debe dejarse sin modificar: ya que Minimax finalmente requerirá considerar muchos estados diferentes de la placa durante su cálculo. Esto significa que simplemente actualizar una celda en la placa en sí no es una implementación correcta de la función de resultado. Es probable que primero desee hacer una copia detallada del tablero antes de realizar cualquier cambio.

 La función de ganador debe aceptar un tablero como entrada y devolver el ganador del tablero si hay uno.

o Si el jugador X ha ganado el juego, su función debe devolver X. Si el jugador O ha

ganó el juego, su función debería devolver O.

o Uno puede ganar el juego con tres de sus movimientos seguidos horizontalmente, verticalmente o diagonalmente.

o Puede suponer que habrá como máximo un ganador (es decir, ningún tablero nunca

tener a ambos jugadores con tres en fila, ya que eso sería un estado de tablero inválido).

o Si no hay un ganador del juego (ya sea porque el juego está en progreso o porque

terminó en un empate), la función debería devolver Ninguno.

 La función de terminal debe aceptar una placa como entrada y devolver un valor booleano

indicando si el juego ha terminado.

o Si el juego termina, ya sea porque alguien ha ganado el juego o porque todas las celdas

se han llenado sin que nadie gane, la función debería devolver True.

o De lo contrario, la función debería devolver False si el juego aún está en progreso.

 La función de utilidad debe aceptar una placa de terminales como entrada y dar salida a la utilidad de la placa.

o Si X ganó el juego, la utilidad es 1. Si O ganó el juego, la utilidad es -1. Si el juego ha terminado en empate, la utilidad es 0.

o Puede asumir que la utilidad solo se llamará en una placa si el terminal (placa) es Verdadero.

 La función minimax debe tomar un tablero como entrada y devolver el movimiento óptimo para que el jugador se mueva en ese tablero.

o El movimiento devuelto debe ser la acción óptima (i, j) que es una de las acciones permitidas en el tablero. Si varios movimientos son igualmente óptimos, cualquiera de esos movimientos es aceptable.

o Si la placa es una placa de terminales, la función minimax debería devolver None.

The player function should take a board state as input, and return which player’s turn it is (either X or O).

* In the initial game state, X gets the first move. Subsequently, the player alternates with each additional move.
* Any return value is acceptable if a terminal board is provided as input (i.e., the game is already over).

The actions function should return a set of all of the possible actions that can be taken on a given board.

* Each action should be represented as a tuple (i, j) where i corresponds to the row of the move (0, 1, or 2) and j corresponds to which cell in the row corresponds to the move (also 0, 1, or 2).
* Possible moves are any cells on the board that do not already have an X or an O in them.
* Any return value is acceptable if a terminal board is provided as input.

The result function takes a board and an action as input, and should return a new board state, without modifying the original board.

* If action is not a valid action for the board, your program should raise an exception.
* The returned board state should be the board that would result from taking the original input board, and letting the player whose turn it is make their move at the cell indicated by the input action.
* Importantly, the original board should be left unmodified: since Minimax will ultimately require considering many different board states during its computation. This means that simply updating a cell in board itself is not a correct implementation of the result function. You’ll likely want to make a deep copy of the board first before making any changes.

The winner function should accept a board as input, and return the winner of the board if there is one.

* If the X player has won the game, your function should return X. If the O player has won the game, your function should return O.
* One can win the game with three of their moves in a row horizontally, vertically, or diagonally.
* You may assume that there will be at most one winner (that is, no board will ever have both players with three-in-a-row, since that would be an invalid board state).
* If there is no winner of the game (either because the game is in progress, or because it ended in a tie), the function should return None.

The terminal function should accept a board as input, and return a boolean value indicating whether the game is over.

* If the game is over, either because someone has won the game or because all cells have been filled without anyone winning, the function should return True.
* Otherwise, the function should return False if the game is still in progress.

The utility function should accept a terminal board as input and output the utility of the board.

* If X has won the game, the utility is 1. If O has won the game, the utility is -1. If the game has ended in a tie, the utility is 0.
* You may assume utility will only be called on a board if terminal(board) is True.

The minimax function should take a board as input, and return the optimal move for the player to move on that board.

* The move returned should be the optimal action (i, j) that is one of the allowable actions on the board. If multiple moves are equally optimal, any of those moves is acceptable.
* If the board is a terminal board, the minimax function should return None.