Implement Tic – tac – toe using Minimax algorithm

```
board = {1: ' ', 2: ' ', 3: ' ',
         4: ' ', 5: ' ', 6: ' ',
         7: ' ', 8: ' ', 9: ' '}

output_printed = False

def printBoard(board):
    global output_printed
    if not output_printed:
        print('Output: 1BM22CS290')
        output_printed = True
    print(board[1] + '|' + board[2] + '|' + board[3])
    print('-+-+-')
    print(board[4] + '|' + board[5] + '|' + board[6])
    print('-+-+-')
    print(board[7] + '|' + board[8] + '|' + board[9])
    print('\n')

def spaceFree(pos):
    return board[pos] == ' '

def checkWin():
    win_conditions = [(1, 2, 3), (4, 5, 6), (7, 8, 9),  # Horizontal
                      (1, 4, 7), (2, 5, 8), (3, 6, 9),  # Vertical
                      (1, 5, 9), (3, 5, 7)]             # Diagonal
    for a, b, c in win_conditions:
        if board[a] == board[b] == board[c] and board[a] != ' ':
            return True
    return False

def checkMoveForWin(move):
    win_conditions = [(1, 2, 3), (4, 5, 6), (7, 8, 9),  # Horizontal
                      (1, 4, 7), (2, 5, 8), (3, 6, 9),  # Vertical
                      (1, 5, 9), (3, 5, 7)]             # Diagonal
    for a, b, c in win_conditions:
        if board[a] == board[b] == board[c] and board[a] == move:
            return True
    return False

def checkDraw():
    return all(board[key] != ' ' for key in board.keys())

def insertLetter(letter, position):
    if spaceFree(position):
        board[position] = letter
        printBoard(board)

        if checkWin():
            if letter == 'X':
                print('Bot wins!')
            else:
                print('You win!')
            return True
        elif checkDraw():
            print('Draw!')
            return True
    else:
        print('Position taken, please pick a different position.')
        position = int(input('Enter new position: '))
        return insertLetter(letter, position)

    return False

player = 'O'
bot = 'X'

def playerMove():
    position = int(input('Enter position for O: '))
    return insertLetter(player, position)

def compMove():
    bestScore = -1000
    bestMove = 0
```

```
        for key in board.keys():
            if board[key] == ' ':
                board[key] = bot
                score = minimax(board, False)
                board[key] = ' '
                if score > bestScore:
                    bestScore = score
                    bestMove = key

        return insertLetter(bot, bestMove)

    def minimax(board, isMaximizing):
        if checkMoveForWin(bot):
            return 1
        elif checkMoveForWin(player):
            return -1
        elif checkDraw():
            return 0

        if isMaximizing:
            bestScore = -1000
            for key in board.keys():
                if board[key] == ' ':
                    board[key] = bot
                    score = minimax(board, False)
                    board[key] = ' '
                    bestScore = max(score, bestScore)
            return bestScore
        else:
            bestScore = 1000
            for key in board.keys():
                if board[key] == ' ':
                    board[key] = player
                    score = minimax(board, True)
                    board[key] = ' '
                    bestScore = min(score, bestScore)
            return bestScore

    game_over = False
    while not game_over:
        game_over = compMove()
        if not game_over:
            game_over = playerMove()
```

```
     | |

     X|X|
     -+-+-
      |O|
     -+-+-
      | |


     Enter position for O: 3
     X|X|O
     -+-+-
      |O|
     -+-+-
      | |


     X|X|O
     -+-+-
      |O|
     -+-+-
     X| |
```

```
-+-+-
X| |


Enter position for O: 7
Position taken, please pick a different position.
Enter new position: 8
X|X|O
-+-+-
O|O|X
-+-+-
X|O|


X|X|O
-+-+-
O|O|X
-+-+-
X|O|X


Draw!
```

Start coding or generate with AI.