In [2]:
```python
from random import choice
from math import inf

board = [[0, 0, 0],
         [0, 0, 0],
         [0, 0, 0]]

def Gameboard(board):
    chars = {1: 'X', -1: 'O', 0: ' '}
    for x in board:
        for y in x:
            ch = chars[y]
            print(f'| {ch} |', end='')
        print('\n' + '---------------')
    print('===============')

def Clearboard(board):
    for x, row in enumerate(board):
        for y, col in enumerate(row):
            board[x][y] = 0

def winningPlayer(board, player):
    conditions = [[board[0][0], board[0][1], board[0][2]],
                  [board[1][0], board[1][1], board[1][2]],
                  [board[2][0], board[2][1], board[2][2]],
                  [board[0][0], board[1][0], board[2][0]],
                  [board[0][1], board[1][1], board[2][1]],
                  [board[0][2], board[1][2], board[2][2]],
                  [board[0][0], board[1][1], board[2][2]],
                  [board[0][2], board[1][1], board[2][0]]]

    if [player, player, player] in conditions:
        return True

    return False

def gameWon(board):
    return winningPlayer(board, 1) or winningPlayer(board, -1)

def printResult(board):
    if winningPlayer(board, 1):
        print('X has won! ' + '\n')

    elif winningPlayer(board, -1):
        print('O\'s have won! ' + '\n')

    else:
        print('Draw' + '\n')

def blanks(board):
    blank = []
    for x, row in enumerate(board):
        for y, col in enumerate(row):
            if board[x][y] == 0:
                blank.append([x, y])

    return blank

def boardFull(board):
    if len(blanks(board)) == 0:
        return True
    return False

def setMove(board, x, y, player):
    board[x][y] = player

def playerMove(board):
    e = True
    moves = {1: [0, 0], 2: [0, 1], 3: [0, 2],
             4: [1, 0], 5: [1, 1], 6: [1, 2],
             7: [2, 0], 8: [2, 1], 9: [2, 2]}
    while e:
        try:
            move = int(input('Enter a number between 1-9: '))
            if move < 1 or move > 9:
                print('Invalid Move! Try again!')
            elif not (moves[move] in blanks(board)):
                print('Invalid Move! Try again!')
            else:
                setMove(board, moves[move][0], moves[move][1], 1)
                Gameboard(board)
                e = False
        except(KeyError, ValueError):
            print('Enter a number!')

def getScore(board):
    if winningPlayer(board, 1):
        return 10

    elif winningPlayer(board, -1):
        return -10

    else:
        return 0

def abminimax(board, depth, alpha, beta, player):
    row = -1
    col = -1
    if depth == 0 or gameWon(board):
        return [row, col, getScore(board)]

    else:
        for cell in blanks(board):
            setMove(board, cell[0], cell[1], player)
            score = abminimax(board, depth - 1, alpha, beta, -player)
            if player == 1:
                # X is always the max player
                if score[2] > alpha:
                    alpha = score[2]
                    row = cell[0]
                    col = cell[1]

            else:
                if score[2] < beta:
                    beta = score[2]
                    row = cell[0]
                    col = cell[1]

            setMove(board, cell[0], cell[1], 0)

            if alpha >= beta:
                break
        if player == 1:
            return [row, col, alpha]

        else:
            return [row, col, beta]

def o_comp(board):
    if len(blanks(board)) == 9:
        x = choice([0, 1, 2])
        y = choice([0, 1, 2])
        setMove(board, x, y, -1)
        Gameboard(board)

    else:
        result = abminimax(board, len(blanks(board)), -inf, inf, -1)
        setMove(board, result[0], result[1], -1)
        Gameboard(board)

def x_comp(board):
    if len(blanks(board)) == 9:
        x = choice([0, 1, 2])
        y = choice([0, 1, 2])
        setMove(board, x, y, 1)
        Gameboard(board)

    else:
        result = abminimax(board, len(blanks(board)), -inf, inf, 1)
        setMove(board, result[0], result[1], 1)
        Gameboard(board)

def makeMove(board, player, mode):
    if mode == 1:
        if player == 1:
            playerMove(board)

        else:
            o_comp(board)
    else:
        if player == 1:
            o_comp(board)
        else:
            x_comp(board)

def pvc():
    while True:
        try:
            order = int(input('Enter to play 1st or 2nd: '))
            if not (order == 1 or order == 2):
                print('Please pick 1 or 2')
            else:
                break
        except(KeyError, ValueError):
            print('Enter a number')

    Clearboard(board)
    if order == 2:
        currentPlayer = -1
    else:
        currentPlayer = 1
```

```python
    while not (boardFull(board) or gameWon(board)):
        makeMove(board, currentPlayer, 1)
        currentPlayer *= -1

    printResult(board)

# Driver Code
print("=================================================")
print("TIC-TAC-TOE using MINIMAX with ALPHA-BETA Pruning")
print("=================================================")
pvc()
```

```
=================================================
TIC-TAC-TOE using MINIMAX with ALPHA-BETA Pruning
=================================================
Enter to play 1st or 2nd: 2
|   || O ||   |
---------------
|   ||   ||   |
---------------
|   ||   ||   |
---------------
===============
Enter a number between 1-9: 5
|   || O ||   |
---------------
|   || X ||   |
---------------
|   ||   ||   |
---------------
===============
| O || O ||   |
---------------
|   || X ||   |
---------------
|   ||   ||   |
---------------
===============
Enter a number between 1-9: 4
| O || O ||   |
---------------
| X || X ||   |
---------------
|   ||   ||   |
---------------
===============
| O || O || O |
---------------
| X || X ||   |
---------------
|   ||   ||   |
---------------
===============
O's have won!
```