Brute Force Approach

In [2]:

```
import random

board = [' ' for x in range(9)]
def main():
    print('Game started')
    print_board()
    game_end = False
    while not game_end:
        print('Player turn')
        player_turn()
        print_board()
        if check_winner(board):
            print('Player won')
            game_end = True
            break

        print('Computer turn')
        computer_move = computer_turn()
        if computer_move != -1:
            board[computer_move] = 'O'
            print_board()
            if check_winner(board):
                print('Computer won')
                game_end = True
                break

        if board.count(' ') < 1:
            print('Tie game')
            game_end = True

    print('Game ended')

def print_board():
    print(board[0] + ' | ' + board[1] + ' | ' + board[2])
    print('---------')
    print(board[3] + ' | ' + board[4] + ' | ' + board[5])
    print('---------')
    print(board[6] + ' | ' + board[7] + ' | ' + board[8])

def check_winner(board):
    if ((board[0] == board[1] == board[2] != ' ') or
```

```python
        (board[3] == board[4] == board[5] != ' ') or
        (board[6] == board[7] == board[8] != ' ')):
        return True

    if ((board[0] == board[3] == board[6] != ' ') or
        (board[1] == board[4] == board[7] != ' ') or
        (board[2] == board[5] == board[8] != ' ')):
        return True

    if ((board[0] == board[4] == board[8] != ' ') or
        (board[2] == board[4] == board[6] != ' ')):
        return True

    return False

def player_turn():
    made_move = False
    while not made_move:
        player_input = input('Enter a position (1-9) ')
        try:
            player_move = int(player_input)
            if player_move < 1 or player_move > 9:
                print('Enter a valid position')
            else:
                player_position = player_move - 1 # player index in board
                if board[player_position] != ' ':
                    print('Position is already taken')
                else:
                    board[player_position] = 'X'
                    made_move = True

        except:
            print('Enter a valid number')


def computer_turn():

    available_moves = [pos for pos, value in enumerate(board) if value == ' ']

    move = -1



    for i in available_moves:
        new_board = board[:]
        new_board[i] = 'O'
        if check_winner(new_board):
            move = i
            return move
    for i in available_moves:
        new_board = board[:]
        new_board[i] = 'X'
        if check_winner(new_board):
            move = i
            return move

    avalable_corners = []
    for i in available_moves:
        if i in [0, 2, 6, 8]:
            avalable_corners.append(i)

    if len(avalable_corners) > 0:
        random_index = random.randrange(0, len(avalable_corners))
        move = avalable_corners[random_index]
        return move

    if 4 in available_moves:
        move = 4
        return move

    avalable_edges = []
    for i in available_moves:
        if i in [1, 3, 5, 7]:
            avalable_edges.append(i)

    if len(avalable_edges) > 0:
        random_index = random.randrange(0, len(avalable_edges))
        move = avalable_edges[random_index]
        return move

    return move

if __name__ == '__main__':

    main()
```

```
Game started
  |   |
---------
  |   |
---------
  |   |
Player turn

  |   |
---------
  | X |
---------
  |   |
Computer turn
```

```
  |   | O
---------
  | X |
---------
  |   |
Player turn

  | X | O
---------
  | X |
---------
  |   |
Computer turn
  | X | O
---------
  | X |
---------
  | O |
Player turn

X | X | O
---------
  | X |
---------
  | O |
Computer turn
X | X | O
---------
  | X |
---------
  | O | O
Player turn

X | X | O
---------
  | X |
---------
X | O | O
Computer turn
X | X | O
---------
  | X | O
---------
X | O | O
Computer won
Game ended
```

Heuristic Approach

In [18]:

```python
import random

class RandomComputerPlayer:
    def __init__(self, letter):
        self.letter = letter

    def get_move(self, game):
        available_moves = game.available_moves()
        return random.choice(available_moves) if available_moves else None

def play(game, x_player, o_player, print_game=True):
    if print_game:
        game.print_board_nums()

    letter = 'X'
    while game.empty_squares():
        if letter == 'O':
            square = o_player.get_move(game)
        else:
            square = x_player.get_move(game)

        if game.make_move(square, letter):
            if print_game:
                print(letter + f' makes a move to square {square}')
                game.print_board()
                print('')   # Empty line

            if game.current_winner:
                if print_game:
                    if game.current_winner == 'O':
                        print('Computer wins!')
                    else:
                        print(letter + ' wins!')
                return game.current_winner

            letter = 'O' if letter == 'X' else 'X'

        # if print_game:
        #     print('It\'s a tie!')
if __name__ == '__main__':
    x_player = HumanPlayer('X')
    o_player = RandomComputerPlayer('O')  # Use the RandomComputerPlayer class here
    t = TicTacToe()
    play(t, x_player, o_player, print_game=True)
```

```
| 0 | 1 | 2 |
| 3 | 4 | 5 |
| 6 | 7 | 8 |
X makes a move to square 8
|   |   |   |
|   |   |   |
|   |   | X |

O makes a move to square 6
|   |   |   |
|   |   |   |
| O |   | X |


X makes a move to square 0
| X |   |   |
|   |   |   |
| O |   | X |

O makes a move to square 5
| X |   |   |
|   |   | O |
| O |   | X |


X makes a move to square 4
| X |   |   |
|   | X | O |
| O |   | X |

X wins!
```

```
In [ ]:
```