**Department of Computer Engineering**

**Academic Term II: 23-24**

**Class: T.E (Computer), Sem – VI Subject Name: Artificial Intelligence Student Name: Roll No: 9538**

| **Practical No:** | **2** |
| --- | --- |
| **Title:** | Tic Tac Toe game implementation by Magic Square Method |
| **Date of Performance:** |  |
| **Date of Submission:** |  |

**Rubrics for Evaluation:**

| **Sr.**  **No** | **Performance Indicator** | **Excellent** | **Good** | **Below**  **Average** | **Marks** |
| --- | --- | --- | --- | --- | --- |
| 1 | On time Completion &  Submission (01) | 01 (On  Time) | NA | 00 (Not on  Time) |  |
| 2 | Logic/Algorithm Complexity analysis (03) | 03(Correct ) | 02(Partial) | 01 (Tried) |  |
| 3 | Coding Standards (03):  Comments/indention/Naming conventions  Test Cases /Output | 03(All  used) | 02 (Partial) | 01 (rarely  followed) |  |
| 4 | Post Lab Assignment (03) | 03(done  well) | 2 (Partially  Correct) | 1(submitte  d) |  |
| **Total** | | | | |  |

**Signature of the Teacher:**

#exp2\_9538\_magicsquare

import random

class TicTacToe:

def \_\_init\_\_(self):

self.board = [' ' for \_ in range(9)] # Representing the Tic-Tac-Toe board

def available\_moves(self):

return [i for i, spot in enumerate(self.board) if spot == ' ']

def empty\_squares(self):

return ' ' in self.board

def num\_empty\_squares(self):

return self.board.count(' ')

def make\_move(self, square, letter):

self.board[square] = letter

def print\_board(self):

for row in [self.board[i \* 3:(i + 1) \* 3] for i in range(3)]:

print('| ' + ' | '.join(row) + ' |')

class Player:

def \_\_init\_\_(self, letter):

self.letter = letter

def get\_move(self, game):

pass

class HumanPlayer(Player):

def get\_move(self, game):

valid\_square = False

val = None

while not valid\_square:

square = input(self.letter + '\'s turn. Input move (0-8): ')

try:

val = int(square)

if val not in game.available\_moves():

raise ValueError

valid\_square = True

except ValueError:

print('Invalid square. Try again.')

return val

class RandomComputerPlayer(Player):

def get\_move(self, game):

square = random.choice(game.available\_moves())

return square

def play(game, x\_player, o\_player, print\_game=True):

if print\_game:

game.print\_board()

letter = 'X' # Starting letter

while game.empty\_squares():

if letter == 'O':

square = o\_player.get\_move(game)

else:

square = x\_player.get\_move(game)

if game.board[square] == ' ':

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.num\_empty\_squares() == 0:

print('It\'s a tie!')

return

if game.winner(square, letter):

print(letter + ' wins!')

return

letter = 'O' if letter == 'X' else 'X' # Switch player

if print\_game:

print('The game is a tie!')

if \_\_name\_\_ == '\_\_main\_\_':

x\_player = HumanPlayer('X')

o\_player = RandomComputerPlayer('O')

t = TicTacToe()

play(t, x\_player, o\_player, print\_game=True)

OUTPUT:

Welcome to Tic Tac Toe!

| |

-------------

| |

-------------

| |

-------------

Enter your move (1-9): 5

| |

-------------

| X |

-------------

| |

-------------

Computer chooses position 8

| |

-------------

| X |

-------------

| O |

-------------

Enter your move (1-9): 7

| |

-------------

| X |

-------------

X | O |

-------------

Computer chooses position 3

| | O

-------------

| X |

-------------

X | O |

-------------

Enter your move (1-9): 1

X | | O

-------------

| X |

-------------

X | O |

-------------

Computer chooses position 4

X | | O

-------------

O | X |

-------------

X | O |

-------------

Enter your move (1-9): 4

ERROR! That position is already taken. Choose a different one.

Enter your move (1-9): 9

X | | O

-------------

O | X |

-------------

X | O | X

-------------

X wins!