Muhammad Waleed
20b-115-se SE-B
AI Lab#04
Sir Nasir Ud Din

Question#01 Simple Reflex Agent (Computer vs Computer)

```
import random
def drawBoard(board):
    #This function prints out the board that is passed to it.
    #"board" is a list of 10 strings representing the board (ignore index
0)
    print()
    print('   |   |')
    print(' '+board[7]+' | ' + board[8]+' | '+board[9])
    print('   |   |')
    print('-----------')
    print(' '+board[4]+' | ' + board[5]+' | '+board[6])
    print('   |   |')
    print('-----------')
    print('   |   |')
    print(' '+board[1]+' | ' + board[2]+' | '+board[3])
    print('   |   |')


def inputPlayerLetter():
    #Lets the player type which letter they want to be their mark
    #Returns a list with the player's letter as the first item, and the
computer's letter as the second.
    #For simplification, keeping X as the player's letter and O as the
computer's letter
    return['X','O']
def whoGoesFirst():
    #for simplification letting the computer go first
    return 'computer'
def playAgain():
    #This function returns True if the player wants to play again,
otherwise it returns False.
    print('Do you want to play again? (yes or no)')
    return input().lower().startswith('y')
def makeMove(board,letter,move):
    #This function simply marks the planned move (Location of the board
with the player's letter.
```

Muhammad Waleed
20b-115-se SE-B
AI Lab#04
Sir Nasir Ud Din

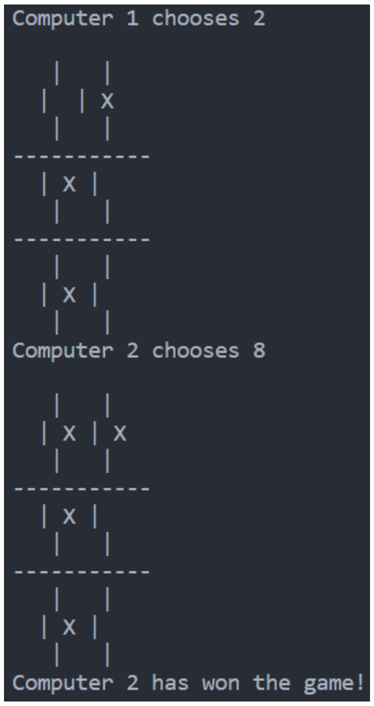
```
    board[move]=letter
def isWinner(bo, le):
    #Given a board and a player's letter, this function returns True if
that player has won.
    #We use bo instead of board and le instead of letter so we don't have
to type as much.
    return ((bo[7]==le and bo[8]==le and bo[9]==le) or # across the top
            (bo[4]==le and bo[5]==le and bo[6]==le) or # across the middle
            (bo[1]==le and bo[2]==le and bo[3]==le) or # across the bottom
            (bo[7]==le and bo[4]==le and bo[1]==le) or #down the left side
            (bo[8]==le and bo[5]==le and bo[2]==le) or #down the middle
            (bo[9]==le and bo[6]==le and bo[3]==le) or #down the right
side
            (bo[7]==le and bo[5]==le and bo[3]==le) or #diagonal
            (bo[9]==le and bo[5]==le and bo[1]==le)) #diagonal
def getBoardCopy(board):
    #Make a duplicate of the board list and return it the duplicate
    dupeBoard=[]
    for i in board:
        dupeBoard.append(i)
    return dupeBoard
def isSpaceFree(board, move):
    # Return true if the passed move is free on the passed board.
    return board[move]==''
def getPlayerMove(board):
    #Let the player type in his move
    move=''
    while move not in '1 2 3 4 5 6 7 8 9'.split() or not
isSpaceFree(board, int(move)):
        print('What is your next move? (1-9)')
        move=input()
    return int(move)
def chooseRandomMoveFromList(board, movesList):
    #Returns a valid move from the passed list on the passed board.
    #Returns None if there is no valid move.
    possibleMoves=[]
    for i in movesList:
        if isSpaceFree(board, i):
```

Muhammad Waleed
20b-115-se SE-B
AI Lab#04
Sir Nasir Ud Din

```
            possibleMoves.append(i)
    if len(possibleMoves)!=0:
        return random.choice(possibleMoves)
    else:
        return None
def getComputerMove(board, computerLetter):
    #Given a board and the computer's letter, determine where to move and
return that move.
    if computerLetter=='X':
        playerLetter='O'
    else:
        playerLetter='X'

    # Here is our algorithm for our tic toc toe AI:
    # First, check if we can win in the next move
    for i in range(1,10):
        copy= getBoardCopy(board)
        if isSpaceFree(copy,i):
            makeMove(copy, computerLetter,i)
            return i
    # Check if the player could win on his next move, and block them.
    for i in range(1,10):
        copy=getBoardCopy(board)
        if isSpaceFree(copy,i):
            makeMove(copy, playerLetter, i)
            if isWinner(copy, playerLetter):
                return i
    # Try to take one of the corners, if they are free
    move = chooseRandomMoveFromList(board, [1,3,7,9])
    if move !=None:
        return move
    #Try to take the center, if it is free.
    if isSpaceFree(board,5):
        return 5
    # Move on one of the sides
    return chooseRandomMoveFromList(board,[2,4,6,8])
def isBoardFull(board):
```

Muhammad Waleed
20b-115-se SE-B
AI Lab#04
Sir Nasir Ud Din

```
    # Return True if every space on the board has been taken. Otherwise
returns False.
    for i in range(1,10):
        if isSpaceFree(board,i):
            return False
    return True
```



```
Computer 1 chooses 2

   |   |
 |  | x
   |   |
-----------
 | x |
   |   |
-----------
   |   |
 | x |
   |   |
Computer 2 chooses 8

   |   |
 | x | x
   |   |
-----------
 | x |
   |   |
-----------
   |   |
 | x |
   |   |
Computer 2 has won the game!
```

Question#01 Lookup Table Approach

```


import random


def drawBoard(board):
    # This function prints out the board that is passed to it.
    # "board" is a list of 10 strings representing the board (ignore index
0)
    print()
```

Muhammad Waleed
20b-115-se SE-B
AI Lab#04
Sir Nasir Ud Din

```python
    print('   |   |')
    print(' '+board[7]+' | ' + board[8]+' | '+board[9])
    print('   |   |')
    print('-----------')
    print(' '+board[4]+' | ' + board[5]+' | '+board[6])
    print('   |   |')
    print('-----------')
    print('   |   |')
    print(' '+board[1]+' | ' + board[2]+' | '+board[3])
    print('   |   |')


def inputPlayerLetter():
    # Lets the player type which letter they want to be their mark
    # Returns a list with the player's letter as the first item, and the
computer's letter as the second.
    # For simplification, keeping X as the player's letter and O as the
computer's letter
    return ['X', 'O']


def whoGoesFirst():
    # for simplification letting the computer go first
    return 'computer'


def playAgain():
    # This function returns True if the player wants to play again,
otherwise it returns False.
    print('Do you want to play again? (yes or no)')
    return input().lower().startswith('y')


def makeMove(board, letter, move):
    # This function simply marks the planned move (Location of the board
with the player's letter.
    board[move] = letter
```

Muhammad Waleed
20b-115-se SE-B
AI Lab#04
Sir Nasir Ud Din

```python
def isWinner(bo, le):
    # Given a board and a player's letter, this function returns True if that player has won.
    # We use bo instead of board and le instead of letter so we don't have to type as much.
    return ((bo[7] == le and bo[8] == le and bo[9] == le) or  # across the top
            (bo[4] == le and bo[5] == le and bo[6] == le) or  # across the middle
            (bo[1] == le and bo[2] == le and bo[3] == le) or  # across the bottom
            (bo[7] == le and bo[4] == le and bo[1] == le) or  # down the left side
            (bo[8] == le and bo[5] == le and bo[2] == le) or  # down the middle
            # down the right side
            (bo[9] == le and bo[6] == le and bo[3] == le) or
            (bo[7] == le and bo[5] == le and bo[3] == le) or  # diagonal
            (bo[9] == le and bo[5] == le and bo[1] == le))  # diagonal


def getBoardCopy(board):
    # Make a duplicate of the board list and return it the duplicate
    dupeBoard = []
    for i in board:
        dupeBoard.append(i)
    return dupeBoard


def isSpaceFree(board, move):
    # Return true if the passed move is free on the passed board.
    return board[move] == ''


def getPlayerMove(board):
    # Let the player type in his move
    move = ''
```

Muhammad Waleed
20b-115-se SE-B
AI Lab#04
Sir Nasir Ud Din

```
    while move not in '1 2 3 4 5 6 7 8 9'.split() or not
isSpaceFree(board, int(move)):
        print('What is your next move? (1-9)')
        move = input()
    return int(move)


def chooseRandomMoveFromList(board, movesList):
    # Returns a valid move from the passed list on the passed board.
    # Returns None if there is no valid move.
    possibleMoves = []
    for i in movesList:
        if isSpaceFree(board, i):
            possibleMoves.append(i)
    if len(possibleMoves) != 0:
        return random.choice(possibleMoves)
    else:
        return None


def getComputerMove(board, computerLetter):
    # Given a board and the computer's letter, determine where to move and
return that move.
    if computerLetter == 'X':
        playerLetter = 'O'
    else:
        playerLetter = 'X'

    # Here is our algorithm for our tic toc toe AI:
    # First, check if we can win in the next move
    for i in range(1, 10):
        copy = getBoardCopy(board)
        if isSpaceFree(copy, i):
            makeMove(copy, computerLetter, i)
            return i
    # Check if the player could win on his next move, and block them.
    for i in range(1, 10):
        copy = getBoardCopy(board)
```

Muhammad Waleed
20b-115-se SE-B
AI Lab#04
Sir Nasir Ud Din

```
        if isSpaceFree(copy, i):
            makeMove(copy, playerLetter, i)
            if isWinner(copy, playerLetter):
                return i
    # Try to take one of the corners, if they are free
    move = chooseRandomMoveFromList(board, [1, 3, 7, 9])
    if move != None:
        return move
    # Try to take the center, if it is free.
    if isSpaceFree(board, 5):
        return 5
    # Move on one of the sides
    return chooseRandomMoveFromList(board, [2, 4, 6, 8])


def isBoardFull(board):
    # Return True if every space on the board has been taken. Otherwise
returns False.
    for i in range(1, 10):
        if isSpaceFree(board, i):
            return False
    return True

def computerVsComputer():
    board = ['']*10
    computer1Letter, computer2Letter = 'X', 'O'
    turn = whoGoesFirst()
    print('The '+turn + ' will go first.')
    play = True

    qTable = {}

    while play:
        if turn == 'computer1':
            state = getState(board, computer1Letter, computer2Letter)

            move = chooseMove(qTable, state)
```

Muhammad Waleed
20b-115-se SE-B
AI Lab#04
Sir Nasir Ud Din

```
            makeMove(board, computer1Letter, move)

            print('Computer 1 has made a move. Board is:')
            drawBoard(board)

            newState = getState(board, computer1Letter, computer2Letter)
            reward = getReward(board, computer1Letter, computer2Letter)

            qTable = updateTable(qTable, state, newState, move, reward)

            if isWinner(board, computer1Letter):
                drawBoard(board)
                print('Computer 1 has won the game!')
                play = False
            else:
                if isBoardFull(board):
                    drawBoard(board)
                    print('The game is a tie!')
                    break
                else:
                    turn = 'computer2'
        else:
            state = getState(board, computer2Letter, computer1Letter)
            move = chooseMove(qTable, state)

            makeMove(board, computer2Letter, move)

            print('Computer 2 has made a move. Board is:')
            drawBoard(board)

            newState = getState(board, computer2Letter, computer1Letter)
            reward = getReward(board, computer2Letter, computer1Letter)

            qTable = updateTable(qTable, state, newState, move, reward)

            if isWinner(board, computer2Letter):
                drawBoard(board)
                print('Computer 2 has won the game!')
```

Muhammad Waleed
20b-115-se SE-B
AI Lab#04
Sir Nasir Ud Din

```python
                play = False
            else:
                if isBoardFull(board):
                    drawBoard(board)
                    print('The game is a tie!')
                    break
                else:
                    turn = 'computer1'



def updateTable(qTable, state, newState, move, reward):
    if state not in qTable:
        qTable[state] = [0, 0, 0, 0, 0, 0, 0, 0, 0]
    if newState not in qTable:
        qTable[newState] = [0, 0, 0, 0, 0, 0, 0, 0, 0]
    qTable[state][move-1] = qTable[state][move-1] + 0.1 * (reward + 0.9 * max(qTable[newState]) - qTable[state][move-1])
    return qTable

def getState(board, computerLetter, playerLetter):
    state = ''
    for i in range(1, 10):
        if board[i] == computerLetter:
            state += '1'
        elif board[i] == playerLetter:
            state += '2'
        else:
            state += '0'
    return state

def getReward(board, computerLetter, playerLetter):
    if isWinner(board, computerLetter):
        reward = 1
    elif isWinner(board, playerLetter):
        reward = -1
    else:
        reward = 0
```

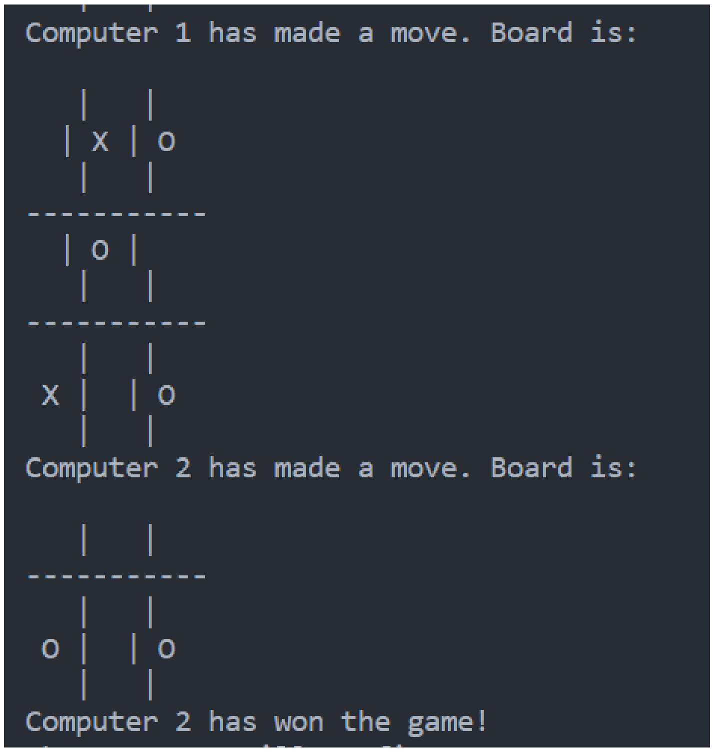
Muhammad Waleed
20b-115-se SE-B
AI Lab#04
Sir Nasir Ud Din

```python
    return reward

def chooseMove(qTable, state):
    if state not in qTable:
        qTable[state] = [0, 0, 0, 0, 0, 0, 0, 0, 0]
    if random.random() < 0.1:
        move = random.randint(1, 9)
    else:
        move = qTable[state].index(max(qTable[state])) + 1
    return move

computerVsComputer()

def computerVsHuman():
    board = ['']*10
    computerLetter, playerLetter = 'X', 'O'
    turn = whoGoesFirst()
    print('The ' + turn + ' will go first.')

    qTable = {}

    while True:
        if turn == 'computer':
            state = getState(board, computerLetter, playerLetter)

            move = chooseMove(qTable, state)

            makeMove(board, computerLetter, move)

            print('Computer has made a move. Board is:')
            drawBoard(board)

            if isWinner(board, computerLetter):
                drawBoard(board)
                print('Computer has won the game!')
                break
            else:
                if isBoardFull(board):
```

Muhammad Waleed
20b-115-se SE-B
AI Lab#04
Sir Nasir Ud Din

```
                    drawBoard(board)
                    print('The game is a tie!')
                    break
                else:
                    turn = 'player'
        else:
            move = getPlayerMove(board)

            makeMove(board, playerLetter, move)

            print('Player has made a move. Board is:')
            drawBoard(board)

            if isWinner(board, playerLetter):
                drawBoard(board)
                print('Player has won the game!')
                break
            else:
                if isBoardFull(board):
                    drawBoard(board)
                    print('The game is a tie!')
                    break
                else:
                    turn = 'computer'


computerVsHuman()
```

Muhammad Waleed
20b-115-se SE-B
AI Lab#04
Sir Nasir Ud Din



```
Computer 1 has made a move. Board is:

   |   |
  | X | O
   |   |
-----------
  | O |
   |   |
-----------
   |   |
 X |   | O
   |   |
Computer 2 has made a move. Board is:

   |   |
-----------
   |   |
 O |   | O
   |   |
Computer 2 has won the game!
```

Question#02: Alter the agent you have written so that it can handle the scenario when the computer goes first or the player/agent goes first.

```
import random


def drawBoard(board):
    # This function prints out the board that is passed to it.
    # "board" is a list of 10 strings representing the board (ignore index
0)
    print()
    print('   |   |')
    print(' '+board[7]+' | ' + board[8]+' | '+board[9])
    print('   |   |')
    print('-----------')
    print(' '+board[4]+' | ' + board[5]+' | '+board[6])
    print('   |   |')
    print('-----------')
    print('   |   |')
```

Muhammad Waleed
20b-115-se SE-B
AI Lab#04
Sir Nasir Ud Din

```
    print(' '+board[1]+' | ' + board[2]+' | '+board[3])
    print('   |   |')


def inputPlayerLetter():
    # Lets the player type which letter they want to be their mark
    # Returns a list with the player's letter as the first item, and the
computer's letter as the second.
    # For simplification, keeping X as the player's letter and O as the
computer's letter
    return ['X', 'O']


def whoGoesFirst():
    # for simplification letting the computer go first
    return 'computer'


def playAgain():
    # This function returns True if the player wants to play again,
otherwise it returns False.
    print('Do you want to play again? (yes or no)')
    return input().lower().startswith('y')


def makeMove(board, letter, move):
    # This function simply marks the planned move (Location of the board
with the player's letter.
    board[move] = letter


def isWinner(bo, le):
    # Given a board and a player's letter, this function returns True if
that player has won.
    # We use bo instead of board and le instead of letter so we don't have
to type as much.
    return ((bo[7] == le and bo[8] == le and bo[9] == le) or  # across the
top
```

Muhammad Waleed
20b-115-se SE-B
AI Lab#04
Sir Nasir Ud Din

```
            (bo[4] == le and bo[5] == le and bo[6] == le) or  # across the
middle
            (bo[1] == le and bo[2] == le and bo[3] == le) or  # across the
bottom
            (bo[7] == le and bo[4] == le and bo[1] == le) or  # down the
left side
            (bo[8] == le and bo[5] == le and bo[2] == le) or  # down the
middle
            # down the right side
            (bo[9] == le and bo[6] == le and bo[3] == le) or
            (bo[7] == le and bo[5] == le and bo[3] == le) or  # diagonal
            (bo[9] == le and bo[5] == le and bo[1] == le))  # diagonal


def getBoardCopy(board):
    # Make a duplicate of the board list and return it the duplicate
    dupeBoard = []
    for i in board:
        dupeBoard.append(i)
    return dupeBoard


def isSpaceFree(board, move):
    # Return true if the passed move is free on the passed board.
    return board[move] == ''


def getPlayerMove(board):
    # Let the player type in his move
    move = ''
    while move not in '1 2 3 4 5 6 7 8 9'.split() or not
isSpaceFree(board, int(move)):
        print('What is your next move? (1-9)')
        move = input()
    return int(move)


def chooseRandomMoveFromList(board, movesList):
```

Muhammad Waleed
20b-115-se SE-B
AI Lab#04
Sir Nasir Ud Din

```python
    possibleMoves = []
    for i in movesList:
        if isSpaceFree(board, i):
            possibleMoves.append(i)
    if len(possibleMoves) != 0:
        return random.choice(possibleMoves)
    else:
        return None


def getComputerMove(board, computerLetter):
    if computerLetter == 'X':
        playerLetter = 'O'
    else:
        playerLetter = 'X'

    for i in range(1, 10):
        copy = getBoardCopy(board)
        if isSpaceFree(copy, i):
            makeMove(copy, computerLetter, i)
            return i
    for i in range(1, 10):
        copy = getBoardCopy(board)
        if isSpaceFree(copy, i):
            makeMove(copy, playerLetter, i)
            if isWinner(copy, playerLetter):
                return i
    move = chooseRandomMoveFromList(board, [1, 3, 7, 9])
    if move != None:
        return move
    if isSpaceFree(board, 5):
        return 5
    return chooseRandomMoveFromList(board, [2, 4, 6, 8])


def isBoardFull(board):
    for i in range(1, 10):
```

Muhammad Waleed
20b-115-se SE-B
AI Lab#04
Sir Nasir Ud Din

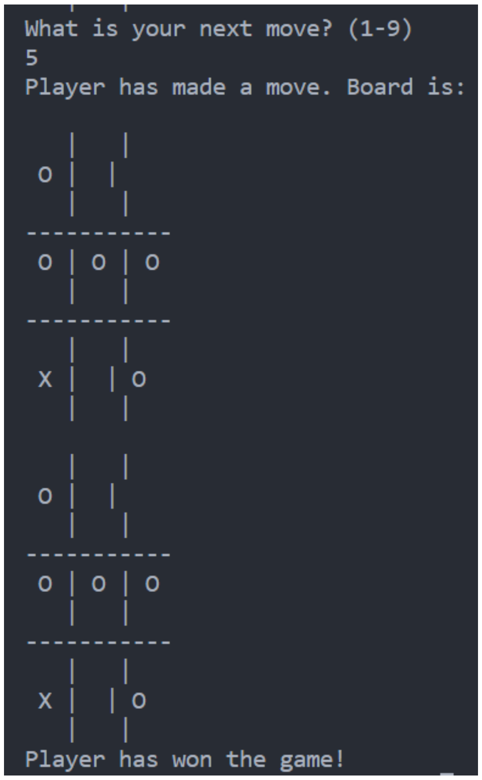
```
        if isSpaceFree(board, i):
            return False
    return True


def getState(board, computerLetter, playerLetter):
    state = ''
    for i in range(1, 10):
        if board[i] == computerLetter:
            state += '1'
        elif board[i] == playerLetter:
            state += '2'
        else:
            state += '0'
    return state


def chooseMove(qTable, state):
    if state not in qTable:
        qTable[state] = [0, 0, 0, 0, 0, 0, 0, 0, 0]
    if random.random() < 0.1:
        move = random.randint(1, 9)
    else:
        move = qTable[state].index(max(qTable[state])) + 1
    return move


def computerVsHuman():
    board = ['']*10
    computerLetter, playerLetter = 'X', 'O'
    turn = whoGoesFirst()
    print('The ' + turn + ' will go first.')

    qTable = {}

    while True:
        if turn == 'computer':
            state = getState(board, computerLetter, playerLetter)
```

Muhammad Waleed
20b-115-se SE-B
AI Lab#04
Sir Nasir Ud Din

```python
            move = chooseMove(qTable, state)

            makeMove(board, computerLetter, move)

            print('Computer has made a move. Board is:')
            drawBoard(board)

            if isWinner(board, computerLetter):
                drawBoard(board)
                print('Computer has won the game!')
                break
            else:
                if isBoardFull(board):
                    drawBoard(board)
                    print('The game is a tie!')
                    break
                else:
                    turn = 'player'
        else:
            move = getPlayerMove(board)

            makeMove(board, playerLetter, move)

            print('Player has made a move. Board is:')
            drawBoard(board)
            if isWinner(board, playerLetter):
                drawBoard(board)
                print('Player has won the game!')
                break
            else:
                if isBoardFull(board):
                    drawBoard(board)
                    print('The game is a tie!')
                    break
                else:
                    turn = 'computer'
```

Muhammad Waleed
20b-115-se SE-B
AI Lab#04
Sir Nasir Ud Din

```
computerVsHuman()
```



```
What is your next move? (1-9)
5
Player has made a move. Board is:

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

   |   |
 O |  |
   |   |
-----------
 O | O | O
   |   |
-----------
   |   |
 X |  | O
   |   |
Player has won the game!
```

Question#03: Alter the agent you have written so that it can handle all the combinations that can be formulated for the 4 cells you have selected.

```
import random



def drawBoard(board):
    # This function prints out the board that is passed to it.
    # "board" is a list of 10 strings representing the board (ignore index
0)
    print()
    print('   |   |')
    print(' '+board[7]+' | ' + board[8]+' | '+board[9])
    print('   |   |')
```

Muhammad Waleed
20b-115-se SE-B
AI Lab#04
Sir Nasir Ud Din

```
    print('-----------')
    print(' '+board[4]+' | ' + board[5]+' | '+board[6])
    print('   |   |')
    print('-----------')
    print('   |   |')
    print(' '+board[1]+' | ' + board[2]+' | '+board[3])
    print('   |   |')


def inputPlayerLetter():
    # Lets the player type which letter they want to be their mark
    # Returns a list with the player's letter as the first item, and the
computer's letter as the second.
    # For simplification, keeping X as the player's letter and O as the
computer's letter
    return ['X', 'O']


def whoGoesFirst():
    # for simplification letting the computer go first
    return 'computer'


def playAgain():
    # This function returns True if the player wants to play again,
otherwise it returns False.
    print('Do you want to play again? (yes or no)')
    return input().lower().startswith('y')


def makeMove(board, letter, move):
    # This function simply marks the planned move (Location of the board
with the player's letter.
    board[move] = letter


def isWinner(bo, le):
```

Muhammad Waleed
20b-115-se SE-B
AI Lab#04
Sir Nasir Ud Din

```
    # Given a board and a player's letter, this function returns True if
that player has won.
    # We use bo instead of board and le instead of letter so we don't have
to type as much.
    return ((bo[7] == le and bo[8] == le and bo[9] == le) or  # across the
top
            (bo[4] == le and bo[5] == le and bo[6] == le) or  # across the
middle
            (bo[1] == le and bo[2] == le and bo[3] == le) or  # across the
bottom
            (bo[7] == le and bo[4] == le and bo[1] == le) or  # down the
left side
            (bo[8] == le and bo[5] == le and bo[2] == le) or  # down the
middle
            # down the right side
            (bo[9] == le and bo[6] == le and bo[3] == le) or
            (bo[7] == le and bo[5] == le and bo[3] == le) or  # diagonal
            (bo[9] == le and bo[5] == le and bo[1] == le))  # diagonal


def getBoardCopy(board):
    # Make a duplicate of the board list and return it the duplicate
    dupeBoard = []
    for i in board:
        dupeBoard.append(i)
    return dupeBoard


def isSpaceFree(board, move):
    # Return true if the passed move is free on the passed board.
    return board[move] == ''


def getPlayerMove(board):
    # Let the player type in his move
    move = ''
    while move not in '1 2 3 4 5 6 7 8 9'.split() or not
isSpaceFree(board, int(move)):
```

Muhammad Waleed
20b-115-se SE-B
AI Lab#04
Sir Nasir Ud Din

```python
        print('What is your next move? (1-9)')
        move = input()
    return int(move)


def chooseRandomMoveFromList(board, movesList):
    # Returns a valid move from the passed list on the passed board.
    # Returns None if there is no valid move.
    possibleMoves = []
    for i in movesList:
        if isSpaceFree(board, i):
            possibleMoves.append(i)
    if len(possibleMoves) != 0:
        return random.choice(possibleMoves)
    else:
        return None


def getComputerMove(board, computerLetter):
    # Given a board and the computer's letter, determine where to move and
return that move.
    if computerLetter == 'X':
        playerLetter = 'O'
    else:
        playerLetter = 'X'

    # Here is our algorithm for our tic toc toe AI:
    # First, check if we can win in the next move
    for i in range(1, 10):
        copy = getBoardCopy(board)
        if isSpaceFree(copy, i):
            makeMove(copy, computerLetter, i)
            return i
    # Check if the player could win on his next move, and block them.
    for i in range(1, 10):
        copy = getBoardCopy(board)
        if isSpaceFree(copy, i):
            makeMove(copy, playerLetter, i)
```

Muhammad Waleed
20b-115-se SE-B
AI Lab#04
Sir Nasir Ud Din

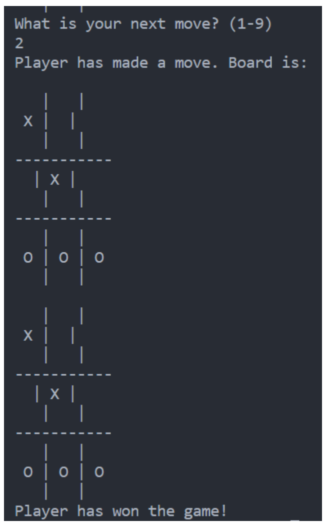
```
            if isWinner(copy, playerLetter):
                return i
    # Try to take one of the corners, if they are free
    move = chooseRandomMoveFromList(board, [1, 3, 7, 9])
    if move != None:
        return move
    # Try to take the center, if it is free.
    if isSpaceFree(board, 5):
        return 5
    # Move on one of the sides
    return chooseRandomMoveFromList(board, [2, 4, 6, 8])


def isBoardFull(board):
    # Return True if every space on the board has been taken. Otherwise
returns False.
    for i in range(1, 10):
        if isSpaceFree(board, i):
            return False
    return True
def getPossibleMoves(board):
    # return a list of all possible moves
    moves = []
    for i in range(1, len(board)):
        if board[i] == '':
            moves.append(i)
    return moves

def getState(board, computerLetter, playerLetter):
    # get the current state
    state = ''
    for i in range(1, 10):
        if board[i] == computerLetter:
            state += '1'
        elif board[i] == playerLetter:
            state += '2'
        else:
            state += '0'
```

Muhammad Waleed
20b-115-se SE-B
AI Lab#04
Sir Nasir Ud Din

```python
    return state
def chooseMove(qTable, state):
    # randomly select a move from the list of possible moves
    if state in qTable:
        possibleMoves = qTable[state]
        move = random.choice(possibleMoves)
    else:
        move = random.randint(1, 9)
    return move

def computerVsHuman():
    board = ['']*10
    computerLetter, playerLetter = 'X', 'O'
    turn = whoGoesFirst()
    print('The ' + turn + ' will go first.')

    qTable = {}

    while True:
        if turn == 'computer':
            state = getState(board, computerLetter, playerLetter)

            move = chooseMove(qTable, state)

            makeMove(board, computerLetter, move)

            print('Computer has made a move. Board is:')
            drawBoard(board)

            if isWinner(board, computerLetter):
                drawBoard(board)
                print('Computer has won the game!')
                break
            else:
                if isBoardFull(board):
                    drawBoard(board)
                    print('The game is a tie!')
                    break
```

Muhammad Waleed
20b-115-se SE-B
AI Lab#04
Sir Nasir Ud Din

```python
                else:
                    turn = 'player'
        else:
            move = getPlayerMove(board)

            makeMove(board, playerLetter, move)

            print('Player has made a move. Board is:')
            drawBoard(board)

            if isWinner(board, playerLetter):
                drawBoard(board)
                print('Player has won the game!')
                break
            else:
                if isBoardFull(board):
                    drawBoard(board)
                    print('The game is a tie!')
                    break
                else:
                    turn = 'computer'


computerVsHuman()
```



```
What is your next move? (1-9)
2
Player has made a move. Board is:

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

   |   |
 X |  |
   |   |
-----------
  | X |
   |   |
-----------
   |   |
 O | O | O
   |   |
Player has won the game!
```