1 Acknowledgment

I respect and thank Dr. Shaheena Noor, for providing me an opportunity to do the project work in Sir Syed University of Engineering University and giving us all support and guidance, which made me complete the project duly. I am extremely thankful to her for providing such a nice support and guidance, although he had busy schedule managing the corporate affairs.

Contents

2 Distribution of Work load:

2.1 Making of Algorithm or logic building:

- Muhammad Taha Ehtesham Khan
- Abrar Ul Eiman

2.2 Writing Program In Python:

- Sheikh Muhammad Adeel

2.3 Making Project Report:

- Muhammad Shoaib

3 Introduction:

Tic-Tac-Toe is a very simple two player game. So only two players can play at a time. This game is also known as Noughts and Crosses or Xs and Os game. One player plays with X and the other player plays with O. In this game we have a board consisting of a 4X4 grid. The number of grids may be increased.

The Tic-Tac-Toe board looks like the following:

Advanced
Tic Tac Toe
4X4 GRID

figure : 1

3.1 Game Rules:

1. Traditionally the first player plays with "X". So you can decide who wants to go with "X" and who wants go with "O".
2. Only one player can play at a time.
3. If any of the players have filled a square then the other player and the same player cannot override that square.
4. There are only two conditions that may match will be draw or may win.
5. The player that succeeds in placing three respective marks (X or O) in a horizontal, vertical or diagonal row wins the game.
6. In this advanced tic-tac-toe game there will be two mode **(easy or normal)**
7. In **"easy mode"** you can win in diagonal by matching 3 rows and column in diagonal or matching in column or rows will not consider as win

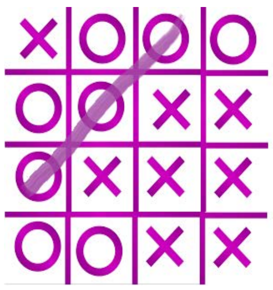

figure : 2

8. In **normal mode** you can play as normal game you would play by matching 4 crosses as you do while playing in 3x3 Matrix

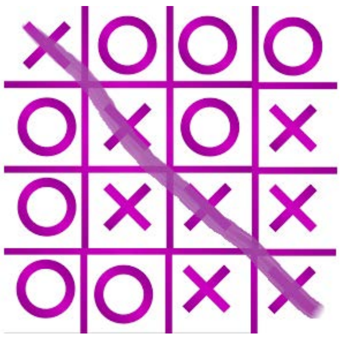

figure: 3

4 Work Flow Diagram

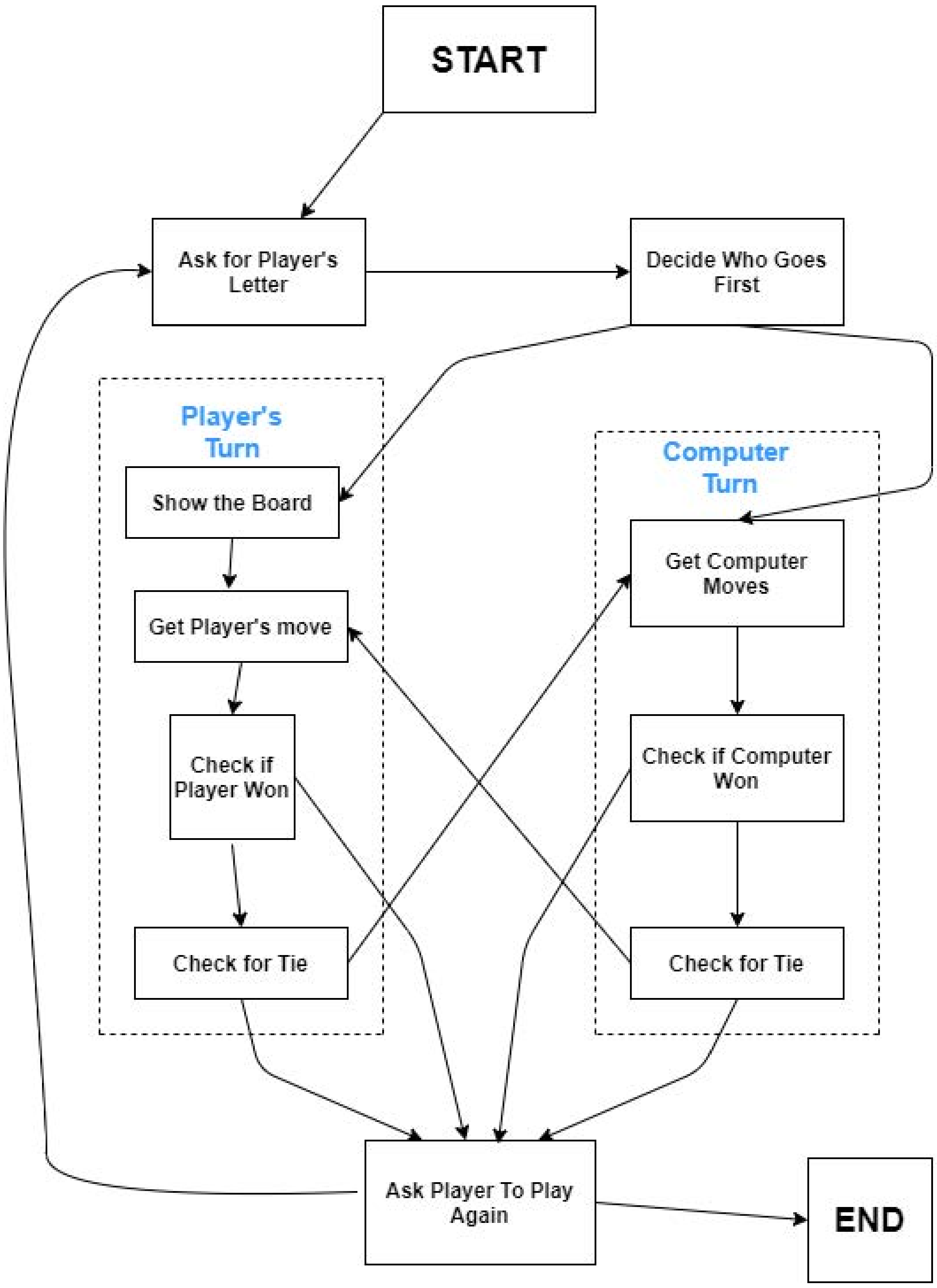

figure : 4

5 Algorithm of Tic-Tac-Toe with AI:

```
step 1: START
step 2: import random library
step 3: drawBoard(board)
        display board
step 4: $easy_easy()$
        print " Do you want to play? (easy or Normal)"
        if input().lower
            return 'easy'
        else
            return 'normal'
step 5: inputPlayerLetter()
        decleare varialbe letter
        initialize letter ← ' '
        do unitl letter will be 'X' or 'O'
            print "Do you want to be 'X' or 'O'?)
            letter ← input().upper
        if letter == 'X'
            return ['X','O']
        else
            return ['O',X']
step 6: whoGoesFirst()
        print "Do you want to go first?(Yes or No)"
        if input().lower
            return 'player'
        else
            return 'computer'
step 7: playAgain()
        print " Do you want to play again? (yes or No) "
            return input().lower()
step 8: makeMove(board,letter,move)
        initialize variable board
        declare board ← letter
step 9: iswiner(board,letter)
        return ((board[1]==letter and board[2]==letter and board[3]==letter
                 and board[4]==letter) or
                (board[3]==letter and board[6]==letter and board[9]==letter)
                or
               (board[8]==letter and board[11]==letter and board[14]==letter)          o
               (board[5]==letter and board[10]==letter and board[15]==letter)
                or
               (board[2]==letter and board[7]==letter and board[12]==letter)
                or
               (board[5]==letter and board[6]==letter and board[7]==letter
                and board[8]==letter) or
               (board[9]==letter and board[10]==letter and board[11]==letter
                and board[12]==letter) or
               (board[13]==letter and board[14]==letter and board[15]==letter
                and board[16]==letter) or
               (board[1]==letter and board[5]==letter and board[9]==letter
                and board[13]==letter) or
               (board[2]==letter and board[6]==letter and board[10]==letter
                and board[14]==letter) or
               (board[3]==letter and board[7]==letter and board[11]==letter
                and board[15]==letter) or
               (board[4]==letter and board[8]==letter and board[12]==letter
                and board[16]==letter) or
               (board[1]==letter and board[6]==letter and board[11]==letter
                and board[16]==letter) or
               (board[4]==letter and board[7]==letter and board[10]==letter
                and board[13]==letter))
step 10: iswinner(board,letter)
        return ((board[1]==letter and board[2]==letter and board[3]==letter
                and board[4]==letter) or
                (board[5]==letter and board[6]==letter and board[7]==letter
                and board[8]==letter) or
                (board[9]==letter and board[10]==letter and board[11]==letter
                and board[12]==letter) or
                (board[13]==letter and board[14]==letter and board[15]==letter
                and board[16]==letter) or
                (board[1]==letter and board[5]==letter and board[9]==letter
                and board[13]==letter) or
                (board[2]==letter and board[6]==letter and board[10]==letter
                and board[14]==letter) or
                (board[3]==letter and board[7]==letter and board[11]==letter
                and board[15]==letter) or
                (board[4]==letter and board[8]==letter and board[12]==letter
                and board[16]==letter) or
                (board[1]==letter and board[6]==letter and board[11]==letter
                and board[16]==letter) or
                (board[4]==letter and board[7]==letter and board[10]==letter
                  and board[13]==letter))
step 11: GetBoardCopy(board)
          declare variable dupBoard
          initialze dupBoard ← []
          for i in board
              dupBoard.append(i)
          return dupBoard
step 12: isSpaceFree(board,move)
          retrun board[move]==' '
          step 13: getPlayerMove(board)
          declare variable move
          initialize move ← ' '
          Move untill '1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16'.split() or not
          isSpaceFree(board,int(move)):
             print " what is your next move?(1-16) "
             move ← input()
          return int(move)
step 13: chooseRandomMoveFromList(board,moveList)
          declare variable possibleMoves
          initialize possibleMoves¡- []
          for i in movesList
             if isSpaceFree(board, i)
                possibleMoves.append(i)
             if len(possibleMoves) ! ← 0
                return random.choice(possibleMoves)
             else
                return None
step 14: minimax(board,depth, isMax, alpha,beta ,computerLetter)
          if computerLetter == 'X':
              playerLetter ← 'O'
          else
              playerLetter ← 'X'
          if isWinner(board, computerLetter)
              return 17
              return 0
          if isMax
                 best ← -1700
                 for i in range(1,10)
                        if isSpaceFree(board, i)
                               board[i] = computerLetter
                               best ← max(best, minimax(board, depth+1, not
                           isMax, alpha, beta,computerLetter) - depth)
                               alpha ← max(alpha, best)
                               board[i] ← ' '
                               if alpha >= beta
                                       break
                return best
          else
                 best = 1700
                 for i in range(1,10)
                           if isSpaceFree(board, i)
                               board[i] = playerLetter
                               best ← min(best, minimax(board, depth+1, not
                           isMax, alpha, beta, computerLetter) + depth)
                               beta ← min(beta, best)
                               board[i] ← ' '
                               if alpha >= beta:
                                       break
                 return best
```
step 15: findBestMove(board, computerLetter)
```
        if computerLetter == 'X'
            playerLetter ← 'O'
        else
            playerLetter ← 'X'
        bestVal ← -1700
        bestMove ← -1
        for i in range(1,17)
                if isSpaceFree(board, i)
                  board[i] ← computerLetter
                  moveVal ← minimax(board, 0, False, -1700, 1700, comput
                erLetter)
                  board[i] ← ' '
                  if moveVal > bestVal:
                      bestMove ← i
                      bestVal ← moveVal
        return bestMove
```
step 16: isBoardFull(board)
```
        for i in range(1,17)
            if isSpaceFree(board, i)
                return False
        return True
```
step 17: print "Welcome to Tic Tac Toe!"

```
          print "Reference of numbering on the board "
          drawBoard('0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16'.split())
          print " " "
step 18: while True:
          theBoard = [' '] * 17
          chose ← $easy_easy()$
          playerLetter, computerLetter = inputPlayerLetter()
          turn ¡- whoGoesFirst()
          print " 'The ' + turn + ' will go first."
          gameIsPlaying ← True
          while gameIsPlaying
                 if turn == 'player'
                    drawBoard(theBoard)
                    move = getPlayerMove(theBoard)
                    makeMove(theBoard, playerLetter, move)
                    if (chose=='easy')
                       if isWiner(theBoard, playerLetter)
                          drawBoard(theBoard)
                          print "You won the game"
                          gameIsPlaying ← False
                       else
                          if isBoardFull(theBoard)
                             drawBoard(theBoard)
                             print "The game is a tie"
                                break
                    else
                               turn ← 'computer'
                else
                             if isWinner(theBoard, playerLetter)
                                drawBoard(theBoard)
                                print "You won the game"
                                gameIsPlaying ← False
                 else
                    if isBoardFull(theBoard)
                       drawBoard(theBoard)
                       print "The game is a tie"
                          break
                    else
                       turn ← 'computer'
                 else
                    move ← findBestMove(theBoard, computerLetter)
```

```
makeMove(theBoard, computerLetter, move)
    if isWinner(theBoard, computerLetter)
        drawBoard(theBoard)
        print "You lose the game"
        gameIsPlaying = False
else
    if isBoardFull(theBoard):
            drawBoard(theBoard)
            print "The game is a tie"
                break
            else
                turn ← 'player'
        if not playAgain()
                break
```

6 Result

```
Welcome to Tic Tac Toe!
Reference of numbering on the board
1  | 2  | 3  | 4
—+—+—+—
5  | 6  | 7  | 8
—+—+—+—
9  | 10| 11| 12
—+—+—+—
13| 14| 15| 16
Do you want to play? (easy or Normal)
Normal
Do you want to be 'X' or 'O'?
O
Do you want to go first? (Yes or No)
No
The computer will go first.
X |    |    |
—+—+—+—
   |    |    |
—+—+—+—
   |    |    |
—+—+—+—
   |    |    |
What is your next move? (1-16)
5
X | X |    |
—+—+—+—
O |    |    |
—+—+—+—
   |    |    |
—+—+—+—
   |    |    |
What is your next move? (1-16)
6
X | X |    |
—+—+—+—
O | O | X |
—+—+—+—
   |    |    |
—+—+—+—
  |   |   |
What is your next move? (1-16)
9
X | X |   |
—+—+—+—
O | O | X | X
—+—+—+—
O |   |   |
—+—+—+—
  |   |   |
What is your next move? (1-16)
10
X | X |   |
—+—+—+—
O | O | X | X
—+—+—+—
O | O | X |
—+—+—+—
  |   |   |
What is your next move? (1-16)
16
X | X |   |
—+—+—+—
O | O | X | X
—+—+—+—
O | O | X | X
—+—+—+—
  |   |   | O
What is your next move? (1-16)
15
X | X |   |
—+—+—+—
O | O | X | X
—+—+—+—
O | O | X | X
—+—+—+—
X |   | O | O
What is your next move? (1-16)
3
X | X | O | X
```

```
—+—+—+—
O | O | X | X
—+—+—+—
O | O | X | X
—+—+—+—
X |   | O | O
What is your next move? (1-16)
13
What is your next move? (1-16)
12
What is your next move? (1-16)
14
X | X | O | X
—+—+—+—
O | O | X | X
—+—+—+—
O | O | X | X
—+—+—+—
X | O | O | O
The game is a tie
Do you want to play again? (Yes or No)
no
```

7 Conclusion:

The Purpose of this project is to implement the knowledge I gain from the whole course which last 4 months, and also learn team management and the distribution of the work and collaboration among team mates, work plan implementation schedule.

8 Reference:

Roopali Garg and Deva Parsad Nayak,2017, Game of Tic-Tac-Toe:Simulation using Min-Max Algorithm,Chandigarh,vol 8,International Journal of Advanced Research in Computer Science