19AIE112 – END SEM PROJECT REPORT

TIC TAC TOE

TEAM - 10

TEAM MEMBERS:

BALAJI M – 21008

G CHAITANYA REDDY – 21014

LOGESHWAR BS – 21025

PINJARI HANEEF – 21034

SRIRAM S - 21054

CONTENTS

INTRODUCTION

TIC TAC TOE Game is commonly known as X O Game which mostly played by our generation guys in our childhood . This game is more of a same kind of game called connect fours which we play using some kind of coins and board , This TIC TAC TOE game is developed version of it . Basically this game requires a data structure to store the values of “X” and “O” like stack . So that’s why we used VM emulator which is build by the stack operation. We play this using a paper and pen by drawing 3 X 3 square table .

This game is a dual player game one is “X” and the other is “O” if any one of theses letters connect it by all the Rows or Columns or Diagonally , Then the player who got it connect is said to be the winner of this game . when all the square are fill without any connect on the same letter then the game is said to drawn. There are almost 255,168 unique ways to play this game out which player who starts first win the game most of the time(131,184 times) and 77,904 times for the player who plays it second and the remaining is draw We will be giving the JACK code and will be running it on the VM emulator after converting the file to .vm file

LITERATURE REVIEW

SIMILAR KIND OF GAME

There are many similar kind of game which play witrh boxes and pen and paper some games are

- Dots and Boxes
- Bingo
- Hangman
- Connect four and etc

This game tic tac toe is made in different languages like Java , Python, Go and etc and. In the other languages we have to use any of the data structure so that the value at a particular position can be saved in a data structure So it makes it a bit difficult to work on .

The changes we made

- We change this game from a multiplayer game to Single player game
- We have changed the background and the board colours to white and black respectively
- Then we have changed few controls also

Methodology

Board.JACK – this code is used to built the board i.e the 3 X 3 box

```
Class Board {
field Array boardArray; // 0 vacant, 1 naught, 2 cross
field int boardX,boardY,boardSize,thickness,cellSize;
field int winThickness; // thickness of the line drawn in case of win!
field boolean isWin, isDraw;
field int winMode; // 0 - row, 1-col, 2 - topdiag, 3 - bottom diag
field int winRow,winCol;// winning row or column
field int freeCellCount; // how many cells are free?


constructor Board new(int x,int y) {
let boardX = x;
let boardY = y;
let boardSize = 160;
let thickness = 4;
let winThickness = 4;
let cellSize = boardSize/3;
let boardArray = Array.new(9);
do clearBoard();
do drawBoardBorders();
return this;
}

method void dispose() {
do Screen.setColor(false);
do Screen.drawRectangle(boardX,boardY,boardX+boardSize, boardY+boardSize);
do Screen.setColor(true);
do Memory.deAlloc(boardArray);
return;
}

method void clearBoard() {
var int i;
let freeCellCount = 9;
let isWin = false;
let isDraw  = false;
while(i<9) {
      let boardArray[i] = 0;
      let i = i + 1;
}
return;
}

// convert row/col to the array index
method int posToIndex(int row, int col) {
return (row*3)+col;
}

// is the selected row/col free of pieces?
method boolean isFree(int row, int col) {
if(boardArray[posToIndex(row,col)]=0) {
      return true;
}else {
      return false;
}
```

```
}

// convert row to pixel y for drawing
method int rowToY(int r) {
return boardY + (r*cellSize);
}

method int colToX(int c) {
return boardX + (c*cellSize);
}

method void drawBoardBorders() {
var int offset;
let offset = thickness/2;
// vertical lines
do                        Screen.drawRectangle(boardX+cellSize-offset,boardY,
boardX+cellSize+offset,boardY+boardSize);
do                    Screen.drawRectangle(boardX+(2*cellSize)-offset,boardY,
boardX+(2*cellSize)+offset,boardY+boardSize);
// horizontal lines
do                        Screen.drawRectangle(boardX,boardY+cellSize-offset,
boardX+boardSize,boardY+cellSize+offset);
do                    Screen.drawRectangle(boardX,boardY+(2*cellSize)-offset,
boardX+boardSize,boardY+(2*cellSize)+offset);
return;
}

// draw win for rows
method void drawRowWin(int r) {
var int y,offset;
let y = rowToY(r)+(cellSize/2);
let offset = winThickness/2;
do Screen.drawRectangle(boardX, y-offset, boardX+boardSize,y+offset);
return;
}

// draw win for columns
method void drawColWin(int c) {
var int x,offset;
let x = colToX(c)+(cellSize/2);
let offset = winThickness/2;
do Screen.drawRectangle(x-offset,boardY, x+offset,boardY+boardSize);
return;
}

// draw diagonal win
method void drawTopDiagonalWin() {
var int i,x,y;
let x = boardX;
let y = boardY;
let i = winThickness-1;
while(~(i<0)) {
      do Screen.drawLine(x+i,y,x+boardSize,y+boardSize-i);
      do Screen.drawLine(x,y+i,x+boardSize-i,y+boardSize);

      let i = i-1;
}
return;
}

// draw diagonal win
method void drawBottomDiagonalWin() {
var int i,x,y;
```

```
let x = boardX;
let y = boardY;
let i = winThickness-1;
while(~(i<0)) {
      do Screen.drawLine(x,y+boardSize-i,x+boardSize-i,y);
      do Screen.drawLine(x+i,y+boardSize,x+boardSize,y+i);

      let i = i-1;
}
return;
}


method void drawWin() {
if(winMode=0) {
      do drawRowWin(winRow);
}
if(winMode=1) {
      do drawColWin(winCol);
}
if(winMode=2) {
      do drawTopDiagonalWin();
}
if(winMode =3) {
      do drawBottomDiagonalWin();
}
return;
}

method boolean isWin() {
return isWin;
}

method boolean isDraw() {
return isDraw;
}

// evaluate board to find draw or win
method void evaluateBoard() {
var int r,c,b1,b2,b3;
let r = 0;
let c = 0;
let isWin = false;
let isDraw = false;

while(r<3) {
      let b1 = boardArray[posToIndex(r,0)];
      let b2 = boardArray[posToIndex(r,1)];
      let b3 = boardArray[posToIndex(r,2)];
      if((b1=b2) & (b2=b3) & (b1>0)) {
            let isWin = true;
            let winMode = 0;
            let winRow = r;
            return;
      }
      let r = r+1;
}
while(c<3) {
      let b1 = boardArray[posToIndex(0,c)];
      let b2 = boardArray[posToIndex(1,c)];
      let b3 = boardArray[posToIndex(2,c)];
      if((b1=b2) & (b2=b3)& (b1>0)) {
            let isWin = true;
            let winMode = 1;
            let winCol = c;
            return;
      }
      let c = c+1;
}

let b1 = boardArray[0];
let b2 = boardArray[4];
let b3 = boardArray[8];
if((b1=b2) & (b2=b3)& (b1>0)) {
      let isWin = true;
      let winMode = 2;
      return;
}

let b1 = boardArray[6];
let b2 = boardArray[4];
let b3 = boardArray[2];
if((b1=b2) & (b2=b3)& (b1>0)) {
      let isWin = true;
      let winMode = 3;
      return;
}

// if no win and also no further moves, we have a draw!
if(freeCellCount=0) {
      let isDraw = true;
      return;
}

return ;
}



method void drawMove(int row, int col,Piece p) {
var int xl,yt;
let xl = colToX(col) + ((cellSize-p.getWidth())/2);
let yt = rowToY(row) + ((cellSize-p.getHeight())/2);
do p.drawPiece(xl,yt);
return;
}

// make a move (doen't draw the move on screen)
method void makeMove(int row, int col, Piece p) {
let boardArray[posToIndex(row,col)] = p.getType();
let freeCellCount = freeCellCount-1;
return;
}

// take back the move
method void takeBackMove(int row, int col,Piece p) {
let boardArray[posToIndex(row,col)] = 0;
let freeCellCount = freeCellCount + 1;
return;
}


method int getFreeCellCount() {
return freeCellCount;
}
```

```
method Array getAvailableMoves() {
var Array moves;
var int r1,c1,counter;
let r1 = 0;
let c1 = 0;
let moves = Array.new(freeCellCount*2);
let counter = 0;
while(r1<3) {
      let c1 = 0;
      while(c1<3) {
            if(boardArray[posToIndex(r1,c1)]=0) {
                  let moves[counter] = r1;
                  let moves[counter+1] = c1;
                  let counter = counter + 2;
            }
            let c1 = c1 + 1;
      }
      let r1 = r1 + 1;
}
return moves;
}
}
```

InputUtil.jack

```
class InputUtil {
static int ir,ic;

function void init() {
let ir = 1;
let ic = 1;
return;
}

function void eraseMessage() {
var String s;
do Output.moveCursor(ir,ic);
let s = "                                                  ";
do Output.printString(s);
do s.dispose();
return;
}

function void showMessage(String s) {
do InputUtil.eraseMessage();
do Output.moveCursor(ir,ic);
do Output.printString(s);
do s.dispose();
return;
}

// reads only valid inputs!
// returns 0 for quit and 1 to 9 for moves!
function int readInput(String s) {
var char c;
var int choice;
do InputUtil.eraseMessage();
while(true) {
      do Output.moveCursor(ir,ic);
      do Output.printString(s);
      let c = Keyboard.readChar();
```

```
        // 48 to 57
        if((c>47) & (c<58)) {
              do s.dispose();
              return c-48;
        }
  }
  return 0;// Never reach here!
  }

  function int waitForZeroOrOne(String s) {
  var char c;
  var int choice;
  do InputUtil.eraseMessage();
  while(true) {
        do Output.moveCursor(ir,ic);
        do Output.printString(s);
        let c = Keyboard.readChar();
        // 48 to 57
        if((c>47) & (c<50)) {
              do s.dispose();
              return c-48;
        }
  }
  return 0;// Never reach here!
  }
  }
```

Main.jack – This code is used to run all the files and restart it again

```
class Main {

      function void main() {
            do Screen.clearScreen();

            do TicTacToeGame.init();
            do TicTacToeGame.startGame();

            return;
      }

}
```

Piece.jack – This code is used to draw “X” and “O”

```
class Piece {
      field int type; // 1 naught, 2 cross
      field int width,height;

      constructor Piece new(int t) {
            let type = t;

            if(t=1) {
                  let width = 20;
                  let height = 20;
            }else {
                  let width = 20;
                  let height = 20;
            }

            return this;
      }

      method int getType() {
            return type;
      }

      method void drawPiece(int x, int y) {
            if(type=1) {
                  do drawNaught(x,y);
            }else {
                  do drawCross(x,y);
            }
            return;
      }

      method void drawNaught(int x, int y) {
            var int ro,ri,t;
            let t = 4;
            let ro = width/2;
            let ri = ro-t;

            do Screen.drawCircle(x+ro,y+ro,ro);
            do Screen.setColor(false);
            do Screen.drawCircle(x+ro,y+ro,ri);
            do Screen.setColor(true);
            return;
      }

      method void drawCross(int x, int y) {
            var int t,i;
            let t = 4;
            let i = t-1;
            while(~(i<0)) {

                  do Screen.drawLine(x,y+height-i,x+width-i,y);
                  do Screen.drawLine(x+i,y+height,x+width,y+i);


                  do Screen.drawLine(x+i,y,x+width,y+height-i);
                  do Screen.drawLine(x,y+i,x+width-i,y+height);

                  let i = i-1;
            }
            return;
      }

      method int getHeight() {
            return height;
      }

      method int getWidth() {
            return width;
      }

}
```

Random.jack

```
class Random {
    static int seed;

    function void setSeed(int newSeed) {
        let seed = newSeed;
        return;
    }



    function int rand() {
        /** return a random number in the range 0..32767 */
        let seed = seed + 20251;
        if (seed < 0) {
            let seed = seed - 32767 - 1;
        }
        return seed;
    }

    function int randRange(int range) {
        /** return a random number in the range 0..range */
        var int mask;
        var int ret;
        let mask = 1;
        while (mask < range) {
            let mask = mask * 2 + 1;
        }
        let ret = Random.rand() & mask;
        while (ret > range) {
            let ret = Random.rand() & mask;
        }
        return ret;
    }

}
```

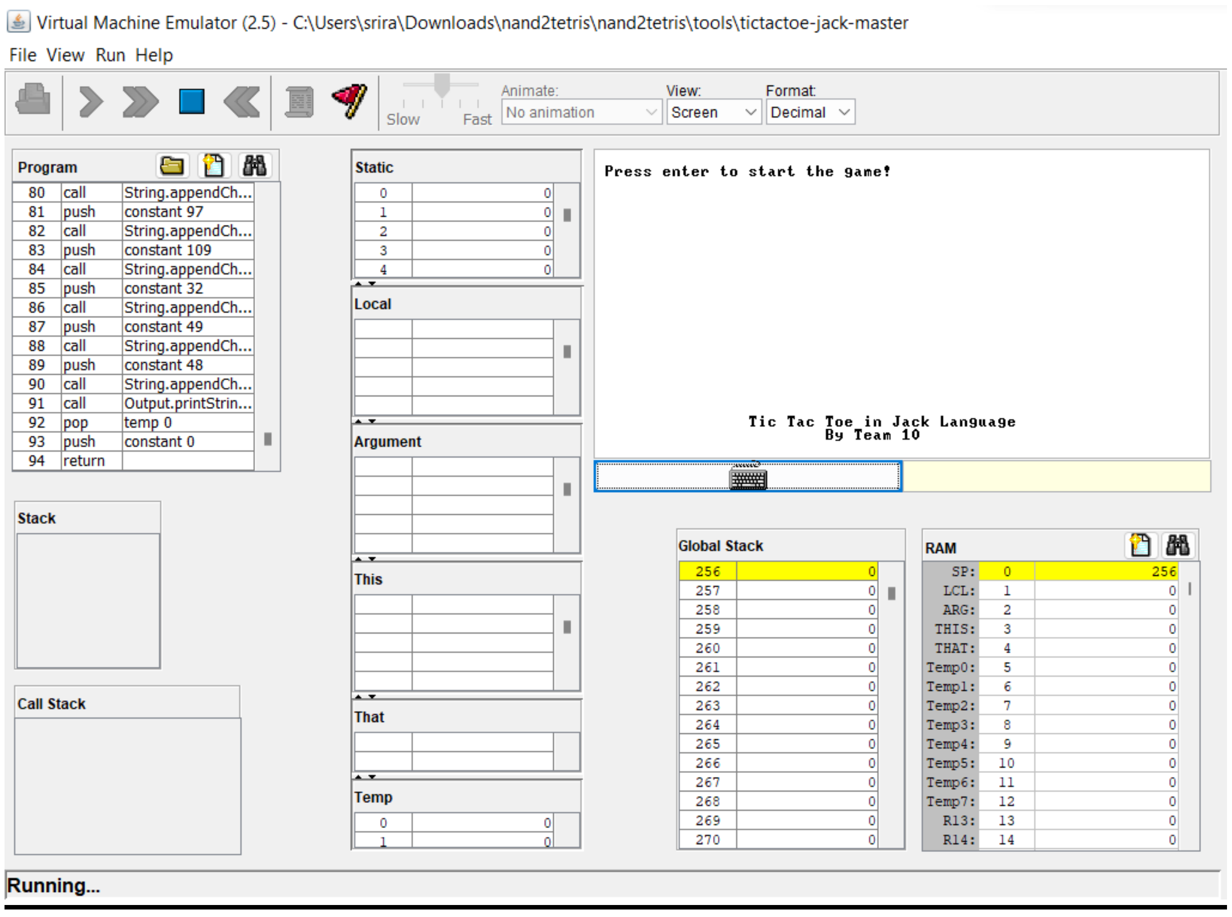
TicTacToeGame.jack – This code is used to get the input and run according to the input

```
class TicTacToeGame {
      static Board board;
      static Piece nPiece,cPiece;
      static int callCounter;
      static boolean isDisplay;
      static int p1;

      function void init() {
            do InputUtil.init();
            let nPiece = Piece.new(1);
            let cPiece = Piece.new(2);
            return;
      }

      function void resetStat() {
            let p1 =0;
            return;
      }
      function void incStat() {
          var String s;
            let p1 = p1+1;
            if(p1=1) {
                do InputUtil.showMessage("Thinking... /");
                  return;
            }
            if(p1=100) {
                  do InputUtil.showMessage("Thinking... -");
                  return;
            }
            if(p1=200) {
                  do InputUtil.showMessage("Thinking... \\");
                  return;
            }
            if(p1=300) {
                  do InputUtil.showMessage("Thinking... |");
                  return;
            }

            if(p1=400) {
                  let p1 = 0;
            }
            return;
      }

      // 1-continue, 2-draw and 3-win;
      function int processBoard() {
            do board.evaluateBoard();
            if(board.isWin()) {
                  do board.drawWin();
            }

            if(board.isWin()) {
                  return 3;
            }
            if(board.isDraw()) {
                  return 2;
            }

            return 1;
      }

      // 0-quit, 1-continue, 2-draw, 3-win
      function int play(int player,Piece p) {
            var boolean validPlayerInput;
            var int input;
            var Array playerMove,computerMove;
            let validPlayerInput = false;

            if(player=0) { // human

                  while(~validPlayerInput) { // the player input must be
                                                                  valid!
                        let input = InputUtil.readInput("Enter move (1 to 9
                                                      or 0 to restart):");
                        if(input=0) {
                              return 0; // restart game!
                        }
                        let playerMove = TicTacToeGame.inputToMove(input);
                        let validPlayerInput =
                                board.isFree(playerMove[0],playerMove[1]);
                        }

                  do board.makeMove(playerMove[0],playerMove[1],p);
                  do board.drawMove(playerMove[0],playerMove[1],p);
                  do playerMove.dispose();
                  return TicTacToeGame.processBoard();
            }else {
                  let computerMove = TicTacToeGame.getComputerMove(p);
                  do board.makeMove(computerMove[0],computerMove[1],p);
                  do board.drawMove(computerMove[0],computerMove[1],p);
                  do computerMove.dispose();
                  return TicTacToeGame.processBoard();
            }
      }

      function void startGame() {
            var int currentPlayer,currentPiece; //  0 for you and 1 for
                                                            computer
            var boolean gameOver;
            var int gameStatus;
            do TicTacToeGame.showCredits();

            do InputUtil.showMessage("Press enter to start the game!");
            do TicTacToeGame.seedGame();

            while(true) {
                  let currentPlayer = InputUtil.waitForZeroOrOne("First
                                      move? (0 for you, 1 for computer): ");
                  let currentPiece = cPiece;

                  let board = Board.new(200,60);
                  let gameOver = false;
                  while(~gameOver) {
                        let gameStatus =
TicTacToeGame.play(currentPlayer,currentPiece);

                        if(gameStatus=0) {
                              let gameOver = true;
                        }
                        if(gameStatus=1) {
                              let currentPlayer =
                              TicTacToeGame.getOtherPlayer(currentPlayer);
                              let currentPiece =
                              TicTacToeGame.getOtherPiece(currentPiece);
                        }
                        if(gameStatus=2) {
                              do InputUtil.showMessage("Game drawn! Press
                                                 enter to restart!");

                              do TicTacToeGame.seedGame();
                              let gameOver = true;
                        }
                        if(gameStatus = 3) {
                              if(currentPlayer=0) {
                                    // this never happens! :-)
                                    do
                                    InputUtil.showMessage("Congratulations!
                                          You won the game!");
                              }else {
                                    do InputUtil.showMessage("Sorry! You
                                lost the game! Press enter to restart!");
                              }
                        do TicTacToeGame.seedGame();
                        let gameOver = true;
                    }
                }
                do board.dispose();// start with a new board!
            }

            return;
        }

        function Array getComputerMove(Piece p) {
            var Array moves;
            var Array movesWithScore;
            do TicTacToeGame.resetStat();
            let moves = Array.new(2);
            if(board.getFreeCellCount()=9) {
                return TicTacToeGame.getRandomGoodMove();
            }else {
                let movesWithScore = TicTacToeGame.getBestMove(p);
                let moves[0]=movesWithScore[1]; //ignore score!
                let moves[1] = movesWithScore[2];
                do movesWithScore.dispose();
                return moves;
            }
        }

        function Array getBestMove(Piece p) {
            var int i,moveCount, currentScore,nextScore;
            var Array moveList, bestMove;

            let i = 0;
            let currentScore = -10; // we always want to get a move!

            let bestMove=Array.new(3);
            let moveCount = board.getFreeCellCount();
            let moveList = board.getAvailableMoves();
            while(i< moveCount) {
                do TicTacToeGame.incStat();
                do board.makeMove(moveList[i+i],moveList[i+i+1],p);
                let nextScore = TicTacToeGame.getMoveScore(p);
                if(nextScore>currentScore) {
                    let currentScore = nextScore;
                    let bestMove[0] = currentScore;
                    let bestMove[1] = moveList[i+i];
                    let bestMove[2] = moveList[i+i+1];
                }
                do board.takeBackMove(moveList[i+i],moveList[i+i+1],p);
                let i = i+1;
            }
            do moveList.dispose();
            return bestMove;
        }

        function int getMoveScore(Piece p) {
            var Array otherMove;
            var int otherScore;
            do board.evaluateBoard();
            if(board.isDraw()) {
                return 0;
            }
            if(board.isWin()) {
                return 1;
            }
            let otherMove=
      TicTacToeGame.getBestMove(TicTacToeGame.getOtherPiece(p));
            let otherScore = otherMove[0];
            do otherMove.dispose();
            return -otherScore;
      }


      function int getOtherPiece(Piece p) {
            if(p = nPiece) {
                  return cPiece;
            }else {
                  return nPiece;
            }
      }

      function int getOtherPlayer(int p) {
            if(p=0) {
                  return 1;
            }else {
                  return 0;
            }
      }

      // get a random corner/center move.
      // used by computer for first move as first move search is very
expensive!
      function Array getRandomGoodMove() {
            var int v;
            var Array moves;
            while(true) {
                  let v = Random.randRange(4);
                  let v = (v*2)+1; // logic for corner pos
                  let moves = TicTacToeGame.inputToMove(v);
                  if(board.isFree(moves[0],moves[1])) {
                        return moves;
                  }
            }
            return moves;

      }

      // get a random move!
      function Array getRandomMove() {
            var int v;
            var Array moves;
            while(true) {
                  let v = Random.randRange(8)+1;
                  let moves = TicTacToeGame.inputToMove(v);
                  if(board.isFree(moves[0],moves[1])) {
                        return moves;
                  }
            }
            return moves;
      }




      function Array inputToMove(int input) {
            var Array moves;
            let moves = Array.new(2);
            let moves[0] = (input-1)/3 ; //row;
```

```
        let moves[1] = (input-1)-(3*moves[0]);
        return moves;

}

function void seedGame() {
        var int seedCounter;
        var boolean enterPressed;
        let enterPressed = false;
        let seedCounter = 1;
        while(~enterPressed) {
                if(Keyboard.keyPressed()=128) {
                        let enterPressed = true;
                }
                let seedCounter = seedCounter+1;
                if(seedCounter > 25444) {
                        let seedCounter = 1;
                }
        }
        do Random.setSeed(seedCounter);
        return;
}

function void showCredits() {
        do Output.moveCursor(20,16);
        do Output.printString("Tic Tac Toe in Jack Language");
        do Output.moveCursor(21,22);
        do Output.printString("By Team 10");
        return;
}
```

Simulation and Result

This how the game initially looks


Virtual Machine Emulator (2.5) - C:\Users\srira\Downloads\nand2tetris\nand2tetris\tools\tictactoe-jack-master
File View Run Help
Slow
Fast
Animate: No animation
View: Screen
Format: Decimal

Program

80	call	String.appendCh...
81	push	constant 97
82	call	String.appendCh...
83	push	constant 109
84	call	String.appendCh...
85	push	constant 32
86	call	String.appendCh...
87	push	constant 49
88	call	String.appendCh...
89	push	constant 48
90	call	String.appendCh...
91	call	Output.printStrin...
92	pop	temp 0
93	push	constant 0
94	return	

Static

0	0
1	0
2	0
3	0
4	0

Local

Argument

This

That

Temp

0	0
1	0

Stack

Call Stack

```
Press enter to start the game!




              Tic Tac Toe in Jack Language
                     By Team 10
```

Global Stack

256	0
257	0
258	0
259	0
260	0
261	0
262	0
263	0
264	0
265	0
266	0
267	0
268	0
269	0
270	0

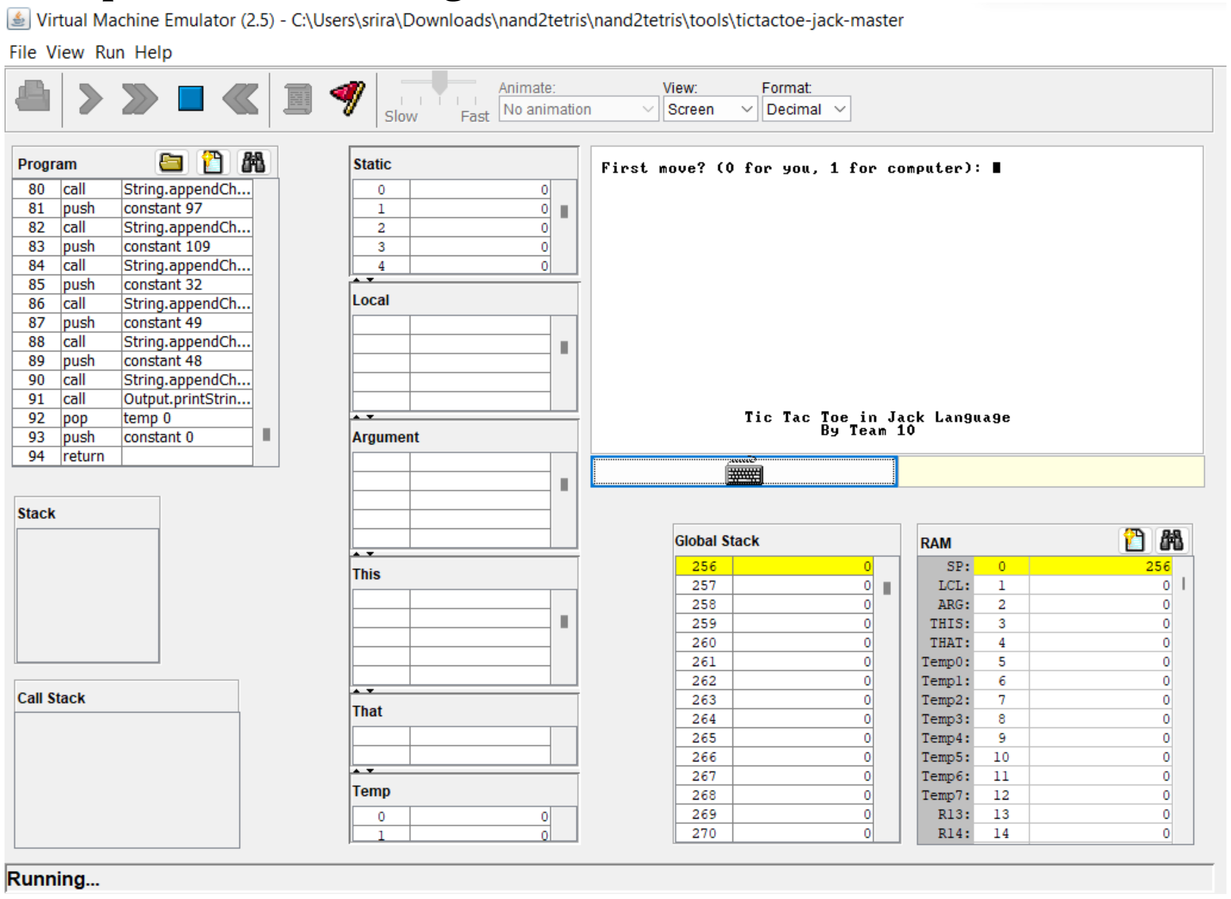
RAM

SP:	0	256
LCL:	1	0
ARG:	2	0
THIS:	3	0
THAT:	4	0
Temp0:	5	0
Temp1:	6	0
Temp2:	7	0
Temp3:	8	0
Temp4:	9	0
Temp5:	10	0
Temp6:	11	0
Temp7:	12	0
R13:	13	0
R14:	14	0

Running...

After entering ENTER it asks us that do you want the first move or computer can do ?

If we press 0 we will be doing the first move and if we press 1 computer will be doing the first move



Animate: No animation   View: Screen   Format: Decimal

Program		
80	call	String.appendCh...
81	push	constant 97
82	call	String.appendCh...
83	push	constant 109
84	call	String.appendCh...
85	push	constant 32
86	call	String.appendCh...
87	push	constant 49
88	call	String.appendCh...
89	push	constant 48
90	call	String.appendCh...
91	call	Output.printStrin...
92	pop	temp 0
93	push	constant 0
94	return	

Static	
0	0
1	0
2	0
3	0
4	0

Temp	
0	0
1	0

```
First move? (0 for you, 1 for computer): ▮




                Tic Tac Toe in Jack Language
                        By Team 10
```

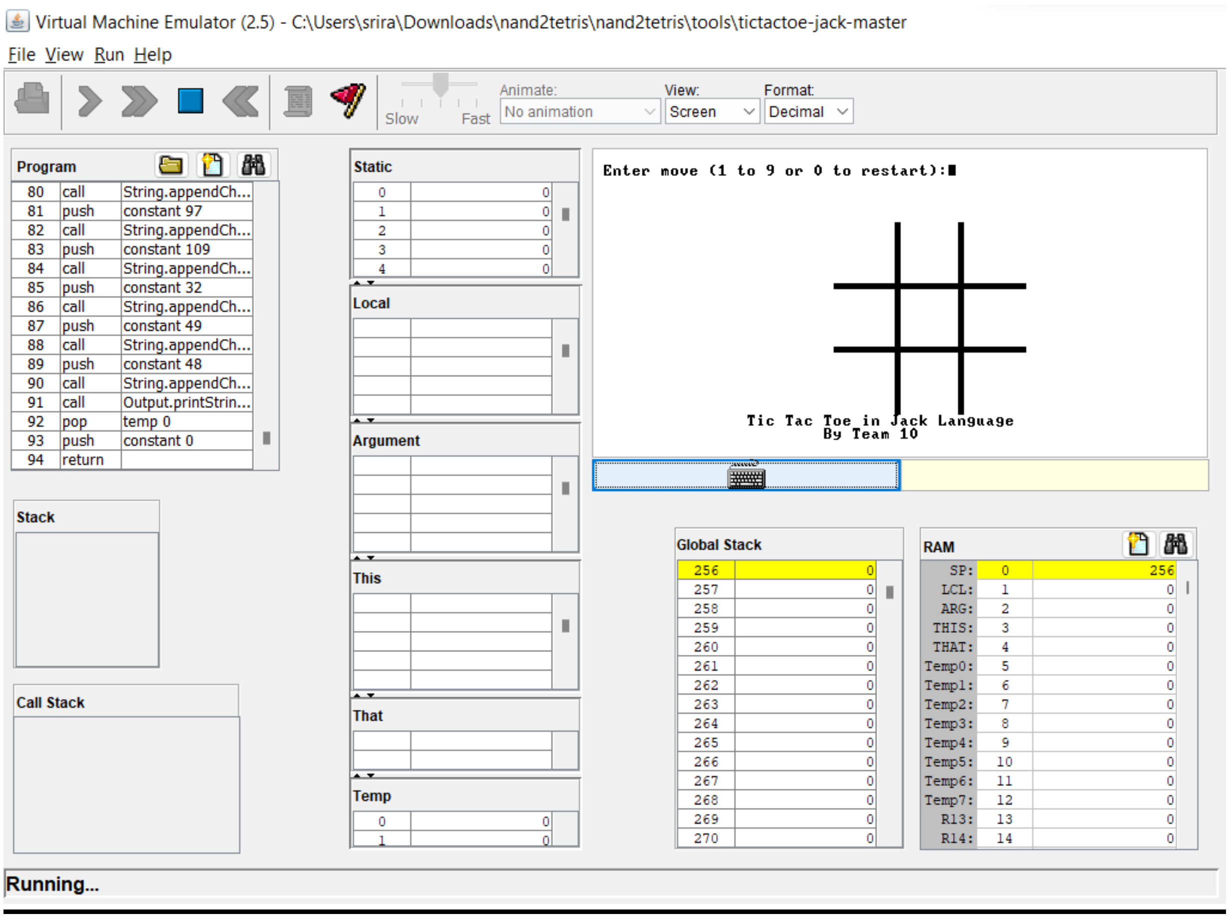
Global Stack	
256	0
257	0
258	0
259	0
260	0
261	0
262	0
263	0
264	0
265	0
266	0
267	0
268	0
269	0
270	0

RAM		
SP:	0	256
LCL:	1	0
ARG:	2	0
THIS:	3	0
THAT:	4	0
Temp0:	5	0
Temp1:	6	0
Temp2:	7	0
Temp3:	8	0
Temp4:	9	0
Temp5:	10	0
Temp6:	11	0
Temp7:	12	0
R13:	13	0
R14:	14	0

Running...

After entering 0 or 1

If we need to fill any boxes we need type the box number between 1 to 9 . If we want to restart the game we need to press 0



Virtual Machine Emulator (2.5) - C:\Users\srira\Downloads\nand2tetris\nand2tetris\tools\tictactoe-jack-master

Animate: No animation
View: Screen
Format: Decimal

Program

80	call	String.appendCh...
81	push	constant 97
82	call	String.appendCh...
83	push	constant 109
84	call	String.appendCh...
85	push	constant 32
86	call	String.appendCh...
87	push	constant 49
88	call	String.appendCh...
89	push	constant 48
90	call	String.appendCh...
91	call	Output.printStrin...
92	pop	temp 0
93	push	constant 0
94	return	

Static

0	0
1	0
2	0
3	0
4	0

Local

Argument

This

That

Temp

0	0
1	0

Stack

Call Stack

```
Enter move (1 to 9 or 0 to restart):

Tic Tac Toe in Jack Language
By Team 10
```

Global Stack

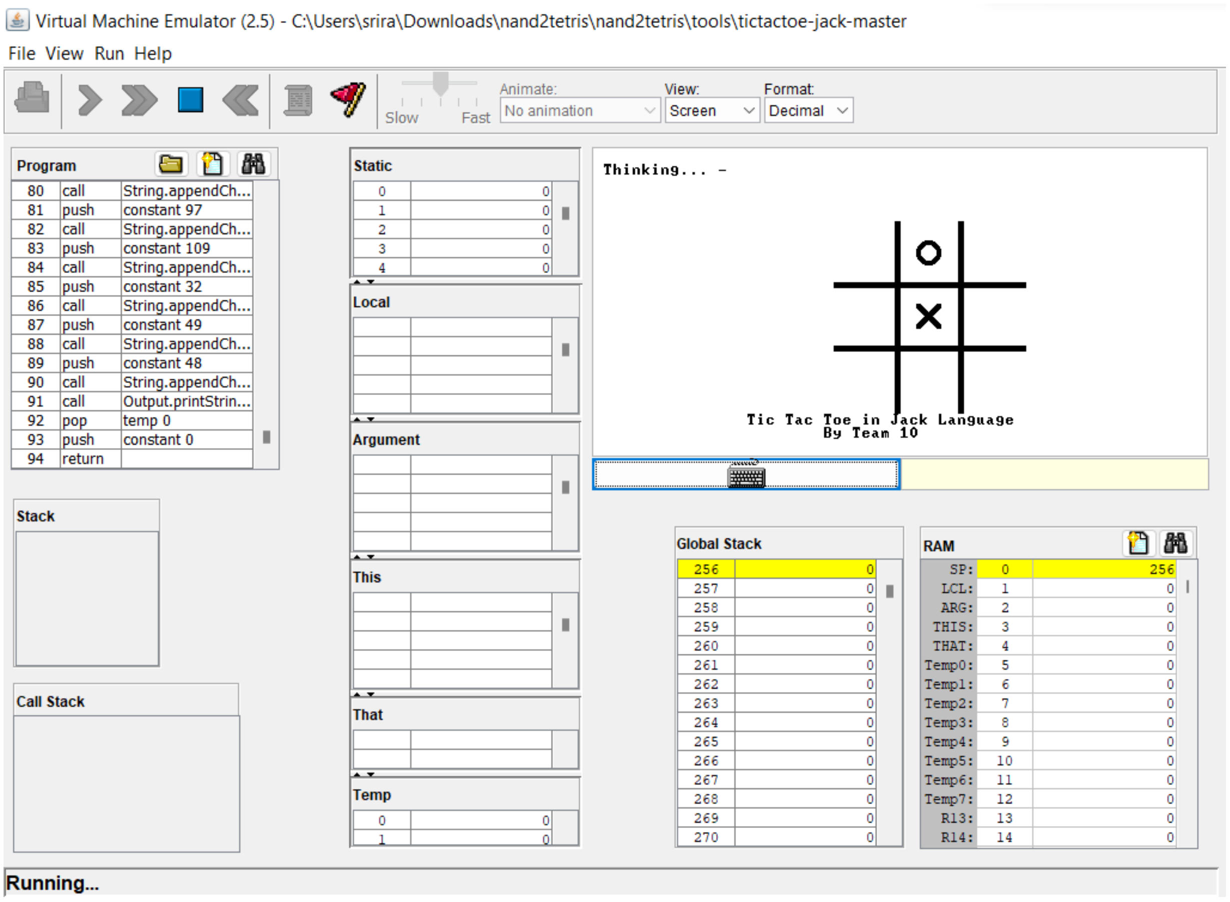
256	0
257	0
258	0
259	0
260	0
261	0
262	0
263	0
264	0
265	0
266	0
267	0
268	0
269	0
270	0

RAM

SP:	0	256
LCL:	1	0
ARG:	2	0
THIS:	3	0
THAT:	4	0
Temp0:	5	0
Temp1:	6	0
Temp2:	7	0
Temp3:	8	0
Temp4:	9	0
Temp5:	10	0
Temp6:	11	0
Temp7:	12	0
R13:	13	0
R14:	14	0

Running...

The Gameplay



Virtual Machine Emulator (2.5) - C:\Users\srira\Downloads\nand2tetris\nand2tetris\tools\tictactoe-jack-master

Animate: No animation
View: Screen
Format: Decimal

Program

```
80  call    String.appendCh...
81  push    constant 97
82  call    String.appendCh...
83  push    constant 109
84  call    String.appendCh...
85  push    constant 32
86  call    String.appendCh...
87  push    constant 49
88  call    String.appendCh...
89  push    constant 48
90  call    String.appendCh...
91  call    Output.printStrin...
92  pop     temp 0
93  push    constant 0
94  return
```

Static

0	0
1	0
2	0
3	0
4	0

Local

Argument

This

That

Temp

0	0
1	0

Stack

Call Stack

```
Thinking... -

Tic Tac Toe in Jack Language
By Team 10
```

Global Stack

256	0
257	0
258	0
259	0
260	0
261	0
262	0
263	0
264	0
265	0
266	0
267	0
268	0
269	0
270	0

RAM

SP:	0	256
LCL:	1	0
ARG:	2	0
THIS:	3	0
THAT:	4	0
Temp0:	5	0
Temp1:	6	0
Temp2:	7	0
Temp3:	8	0
Temp4:	9	0
Temp5:	10	0
Temp6:	11	0
Temp7:	12	0
R13:	13	0
R14:	14	0

Running...

Conclusion

The player who succeeded in placing three respective marks in horizontal, vertical , or diagonal row wins the game . Tic Tac Toe is great way to pass free time whether you’re standing in a line or spending time with your kids

Future Implementation]

- We can add some more colours which may look colourful
- We can do the game called connect four by making 5X5 and to connect 4

References

https://devpost.com/software/tictactoe-game-using-jack-language

https://www.google.com/ for some definitions