**A PROJECT REPORT ON**

**ELEMENTS OF COMPUTING – 2**

**“TICTACTOE GAME USING JACK LANGUAGE”**

**Submitted by**

SANGGIT SAARAN K C S (CB.SC.U4AIE23247)

SURYA HA (CB.SC.U4AIE23267)

VISHAL SESHADRI B (CB.SC.U4AIE23260)

VENKATRAM KS (CB.SC.U4AIE23236)

In partial fulfilment for the award of the degree of BACHELOR OF TECHNOLOGY CSE(AI)

Under the Guidance of **B. Ganga Gowri** Centre for Computational Engineering and Networking AMRITA SCHOOL OF ARTIFICIAL INTELLIGENCE AMRITA VISHWA VIDYAPEETHAM COIMBATORE - 641 112 (INDIA)

**ACKNOWLEDGEMENT**

We would like to express our sincere gratitude to Dr. K P Soman, Dean of our Department, for providing us the opportunity to work on the project entitled “**TICTACTOE GAME USING JACK LANGUAGE” ,** is the record of the original work done by our team underthe guidance of B. Ganga Gowri Assistant professor, Centre Computational Engineering and Networking Amrita school of Artificial Intelligence, Coimbatore. To the best of our knowledge this work has not formed the basis for the award of any degree/diploma/associate ship/fellowship/or a similar award to any candidate in any university.

**Signature of the Faculty:**

|  |  |  |
| --- | --- | --- |
| **Project Team:** | |  |
|  | **Name** | **(Roll Number)** |
| 1. | SANGGIT SAARAN K C S | (CB.SC.U4AIE23247) |
| 2. | VISHAL SESHADRI B | (CB.SC.U4AIE23260) |
| 3. | SURYA H A | (CB.SC.U4AIE23267) |
| 4. | VENKATRAM K S | (CB.SC.U4AIE23236) |

**Place**: Coimbatore

**Date:** 03-06-2024

**TABLE OF CONTENTS**

**CONTENTS** **PAGE NO.**

1. Abstract……………………………….…………………………………………….4
2. Introduction……………………………………………….……….….….………...5
3. Literature review……………………..……..………………….….………………..6
4. Design and Implementation.………………..…………..……..…………….….…...8
5. Testing and evaluation …………………………………...…...…..….....…….…...12
6. Results and Discussion……………………………………..….….……..…...……20
7. Conclusions……………………………………………….……….……….……....21
8. References…………………….……………………………………………..……22

1. **ABSTRACT**

In the realm of computer architecture education, implementing theoretical concepts with practical applications is crucial. This paper focuses on the implementation of the Tic Tac Toe game using the jack language and Nand2Tetris framework. By utilising Nand2tetris framework, this project showcases the creation of a fully functional software application.

Through extensive testing and evaluation, our Tic-Tac-Toe implementation demonstrated robust performance, accurate game logic, and an intuitive user experience. The results showcase how foundational knowledge can be applied to create engaging, interactive software, aiming to inspire further research and collaboration in computer architecture education. This paper highlights the potential of learning through project-based, to enrich the computer architecture education.

1. **INTRODUCTION**

The enduring game of Tic-Tac-Toe, also known as Noughts and Crosses, has captivated players of all ages for generations. Despite its simple rules and gameplay, this classic game provides a rich avenue for exploring various aspects of computer science, such as algorithm design, data structures, and user interfaces. The goal of this project is to create a fully functional, interactive Tic-Tac-Toe game using the Jack language, a sophisticated, object-oriented programming language specifically designed for the Nand2Tetris software.

At the heart of modern computing lies computer architecture, which involves the complex design and integration of hardware and software components that drive computational devices. Educators and researchers continuously seek effective methods to teach these intricate concepts, aiming to give students a thorough understanding of theoretical foundations and practical applications. The Nand2Tetris software suite stands out as a comprehensive educational platform presenting a distinctive approach to comprehending computer systems from their foundational roots. The Jack language plays a pivotal role within this suite, offering a contemporary, object-oriented programming language that facilitates the development of interactive applications.

The objectives of this term work are multi-faceted. Firstly, it aims to implement the Tic-Tac-Toe game in the Jack language as a practical exercise in applying object-oriented programming principles, including encapsulation, inheritance, and polymorphism. Secondly, it seeks to explore efficient algorithms for assessing game states, different game boards, detecting winning conditions, and managing draw scenarios. Lastly, the project is designed to ensure that these features are accessible for future modifications and extensions, promoting an ongoing learning process.

The following sections of this report are organized as follows: Section 2 reviews related work and existing implementations of Tic-Tac-Toe games. Section 3 details the design and implementation aspects of our project, including the overall architecture, key components, and algorithms used. Section 4 describes the testing and evaluation methodology, along with the results obtained. Section 5 provides a detailed discussion of the project's strengths, limitations, and potential for future improvements. Finally, Section 6 concludes the report by summarizing the main achievements and contributions of this project.

1. **RELATED WORK**

The integration of computer architecture education with practical software application development has been explored through various teaching methodologies. This section reviews relevant literature on educational approaches and techniques related to our work, focusing on the Nand2Tetris project, project-based learning (PBL) in computer architecture, and the development of educational software using high-level programming languages.

The Nand2Tetris project, an extensive educational initiative created by Noam Nisan and Shimon Schocken, aims to demystify computer architecture by guiding students through the process of building a fully functional computer from basic NAND gates to a high-level language and operating system. The project's success lies in its immersive method, which allows students to thoroughly explore both the hardware and software aspects of computer architecture. Previous studies have confirmed the effectiveness of Nand2Tetris in improving students' understanding of complex computer architecture concepts. Notably, Schocken and Nisan (2008) found that students who completed the Nand2Tetris course showed significant improvement in their comprehension of hardware-software integration and system design.

Project-based learning (PBL) is widely recognized as an effective teaching strategy in computer science education, promoting active learning through real-world projects, enhancing their learning experience. For example, research by Smith and Goodwin (2010) revealed that students in project-based computer architecture courses developed better problem-solving skills and a deeper understanding of system-level design compared to those in traditional lecture-based courses.

Incorporating high-level programming languages in educational software development projects provides students with hands-on experience in software engineering and system design. The Jack programming language, integral to the Nand2Tetris curriculum, is a simplified high-level object-oriented programming language and software development tool. Research on the educational benefits of creating software applications, such as games, in educational settings has shown that these projects can significantly enhance students' coding skills and creativity. A study by Garcia et al. (2013) examined the impact of game development projects on students' learning outcomes and found that such projects increased students' motivation and engagement, leading to better retention of programming concepts. Developing games like Tic-Tac-Toe using educational tools like Jack offers a structured yet flexible framework for students to experiment with software design and logical implementation.

A notable implementation is by Jayson Joseph, titled TicTacToe.jack. This version features an AI opponent, allowing for player vs. computer gameplay, in contrast to our project which focuses on player vs. player interactions. Joseph’s game is limited to a 3x3 board and does not include an undo feature. While Joseph’s project excels in providing a challenging single-player experience, our project offers more flexibility with additional board sizes and an undo feature, improving usability and user experience. Despite these differences, both projects aim to utilize the Jack programming language within the Nand2Tetris framework to create educational and engaging applications.

The literature highlights the effectiveness of integrated educational frameworks like Nand2Tetris in teaching computer architecture through a project-based approach. By combining hardware design principles with high-level programming, these initiatives provide a comprehensive learning experience that bridges theoretical concepts with practical application. Creating a Tic-Tac-Toe game using the Jack language within the Nand2Tetris framework is a novel contribution to this educational paradigm, demonstrating the potential of project-based learning in fostering a deeper understanding of computer architecture.

1. **DESIGN AND IMPLEMENTATION**

**Board.jack :**

The **Board** class in our Tic-Tac-Toe implementation manages the game state, handles player moves including undo functionality, evaluates game states for wins or draws, and renders the board and moves on the screen. The class features several key variables such as **boardArray** to store the state of the game board, **boardX**, **boardY**, **boardSize**, **thickness**, and **cellSize** for the board's dimensions and drawing parameters, and **winThickness** for the thickness of the winning line. Boolean flags **isWin** and **isDraw** indicate if the game has been won or drawn, respectively. Variables **winMode**, **winRow**, and **winCol** store information about the winning mode and the specific row or column. **freeCellCount** tracks the number of vacant cells, while **dimension** represents the board's size. The **moveStack** array records move for undo functionality, with **top** representing the stack's top index. The class also includes an instance of the **Piece** class for drawing moves, with **pieceWidth** and **pieceHeight** defining the game pieces’ dimensions.

The constructor **Board(int x, int y, int dim)** initializes the board at the specified coordinates and dimension, calculating board size and cell size, initializing **boardArray**, clearing the board, and drawing board borders. Methods include **dispose**(), which clears the board area and deallocates memory, and **clearBoard**(), which resets the game state. The **posToIndex(int row, int col, int dim)** method converts row and column coordinates to an array index, while **isFree(int row, int col)** checks if a cell is vacant. Drawing methods like **drawBoardBorders**(), **drawRowWin**(**int r**), **drawColWin**(**int c**), **drawTopDiagonalWin**(), and **drawBottomDiagonalWin**() handle rendering the board and winning lines. The **drawWin**() method invokes the appropriate win drawing method based on **winMode**.

Game state evaluation methods include **isWin**() and **isDraw**(), with **evaluateBoard**() assessing the board for wins or draws and updating relevant variables. The **drawMove(int row, int col, Piece p)** method visually places a move on the board, while **makeMove(int row, int col, Piece p)** updates the game state with the move, evaluates the board, and records the move in **moveStack**. The **takeBackMove**() method undoes the last move, restoring the previous state and reevaluating the board. The helper method **eraseMove(int row, int col)** assists with undo functionality by removing the move from the screen and redrawing borders. Additionally, **hasMoves**() checks for available undo moves, **getFreeCellCount**() returns the number of vacant cells, and **getAvailableMoves**() provides coordinates of all free cells. This comprehensive `Board` class serves as the core logic and data structure for the Tic-Tac-Toe game, ensuring robust management of game states and user interactions.

**TicTacToeGame.jack :**

The **TicTacToeGame** class acts as the primary controller for the Tic-Tac-Toe game, managing game initialization, player input, game logic, and overall game flow. Key class variables include an instance of the **Board** class to represent the game board, instances of the **Piece** class (**nPiece** and **cPiece**) for the two players, and variables like **seed**, and **dimension**. seed stores the seed value for randomization, and dimension holds the board size (3, 4, or 5).

The **init**() method sets up the game by initializing the **InputUtil** class and creating instances of **nPiece** and **cPiece**. Placeholder methods **resetStat**() and **incStat**() are present for future use. The **processBoard**() method evaluates the board's current state, calls the **evaluateBoard**() method on the board instance, and, if there's a win, draws the winning line, returning an integer to indicate the game state (1 for continue, 2 for draw, 3 for win). The **play(int player, Piece p)** method manages player moves by prompting for input, validating it, and making the move on the board using the **makeMove**() method. It also supports undo functionality and returns an integer based on the game state (0 for quit, 1 for continue, 2 for draw, 3 for win).

The main game loop is handled by the **startGame**() method, which displays game credits, prompts for board dimension, initializes the board, and alternates between players until the game ends (win, draw, or quit). It manages game over conditions and restarts the game if chosen by the user. Additional utility methods include **getOtherPiece(Piece p)** and **getOtherPlayer(int p)** for switching between players, **inputToMove(int input)** for converting user input to board coordinates, **setSeed(int newSeed)** for setting the seed value, **seedGame**() for generating a random seed based on time, and **showCredits**() for displaying game credits. Overall, the TicTacToeGame class integrates with the `Board` and `Piece` classes to manage the game state, player interactions, and rendering, making it the core of the game's control logic.

**InputUtil.jack :**

The **InputUtil** class serves as a utility for managing user input and screen messages in the Tic-Tac-Toe game. It features static integer variables **ir** and **ic** to represent the row and column coordinates for message display. The **init**() method initializes these coordinates to 1. The **eraseMessage**() method clears the previous message by positioning the cursor at ir and ic and printing a space character. The **showMessage(String s)** method first calls **eraseMessage**(), then displays the string **s** at the coordinates. The **readInt(String prompt)** method displays a prompt and waits for the user to enter a single digit, reading characters until a valid digit is entered, which is then returned as an integer. This method is crucial for obtaining integer inputs from users. The **readInput3(String s)**, **readInput4(String s)**, and **readInput5(String s)** methods are tailored for different board sizes (3x3, 4x4, and 5x5), allowing inputs within the valid range for each board size, as well as **0** to quit and **U** to undo the last move. Each of these methods disposes of the input string and returns the appropriate value or -1 for undo. The **waitForZeroOrOne(String s)** method waits for the user to input either **0** or **1** for choosing the player's piece (0 for naught, 1 for cross) and returns the corresponding integer. The InputUtil class simplifies input handling by ensuring valid moves and managing message display, interacting with the screen and keyboard through the **Output** and **Keyboard** classes of the Jack language. This class ensures efficient user interaction by overwriting old messages with new ones at consistent screen locations.

**Piece.jack :**

The **Piece** class is essential in representing and rendering the game pieces (naughts and crosses) for the Tic-Tac-Toe game. This class includes fields such as **type**, which indicates whether the piece is a naught (1) or a cross (2), and **width** and **height**, both set to a default value of 16, to define the piece's dimensions. The constructor **Piece(int t)** initializes the type based on the provided parameter **t**.

The **getType**() method returns the type of the piece, allowing the game logic to differentiate between naughts and crosses. The **drawPiece(int x, int y)** method is responsible for rendering the piece on the screen at specified coordinates (x, y). Depending on the piece type, it invokes either **drawNaught** or **drawCross**. The **drawNaught(int x, int y)** method draws a circular naught by calculating outer and inner radii based on the piece's dimensions and using **Screen.drawCircle** to create a ring effect. Conversely, the **drawCross(int x, int y)** method constructs a cross by drawing four intersecting lines within the piece's dimensions using the **Screen.drawLine** method. Additionally, **getHeight**() and **getWidth**() methods return the height and width of the piece, respectively, aiding in precise positioning and rendering. The `Piece` class encapsulates the functionality for creating and drawing game pieces on the screen, ensuring that the visual representation of naughts and crosses is handled efficiently. It integrates seamlessly with the **Screen** class from the Jack language, playing a pivotal role in the visual aspect of the Tic-Tac-Toe game.

**Main. Jack :**

The **Main** class serves as the entry point for initializing and starting the Tic-Tac-Toe game. It consists of a single method, **main**(), which orchestrates the game's setup and execution. The method begins by clearing the screen using **Screen.clearScreen**(), ensuring that any previous output does not interfere with the game's visuals. Next, it calls **TicTacToeGame.init**(), which sets up the necessary game components and configurations. Finally, the method invokes **TicTacToeGame.startGame**(), which initiates the main game loop, allowing players to start playing Tic-Tac-Toe. This straightforward sequence ensures that the game is ready to be played immediately after the program is run.

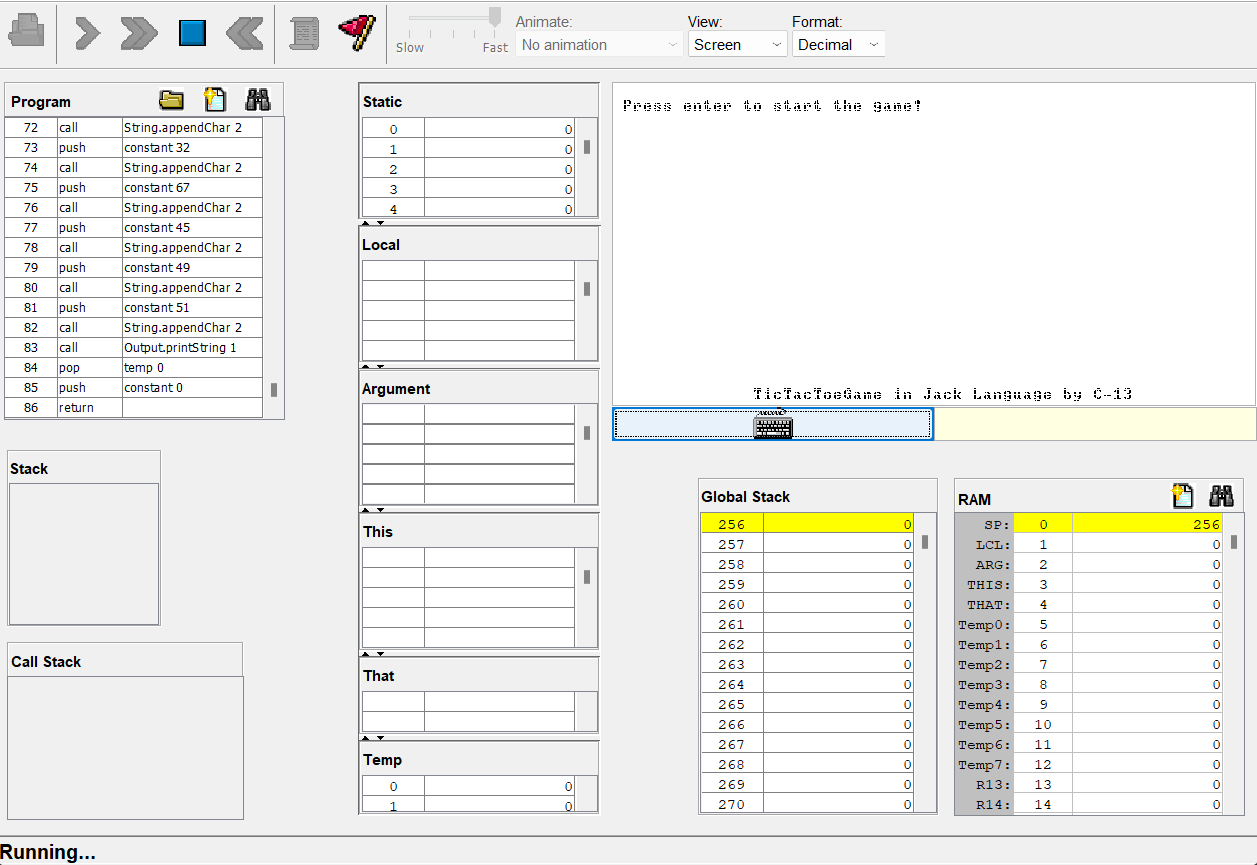
1. **TESTING AND EVALUATION**

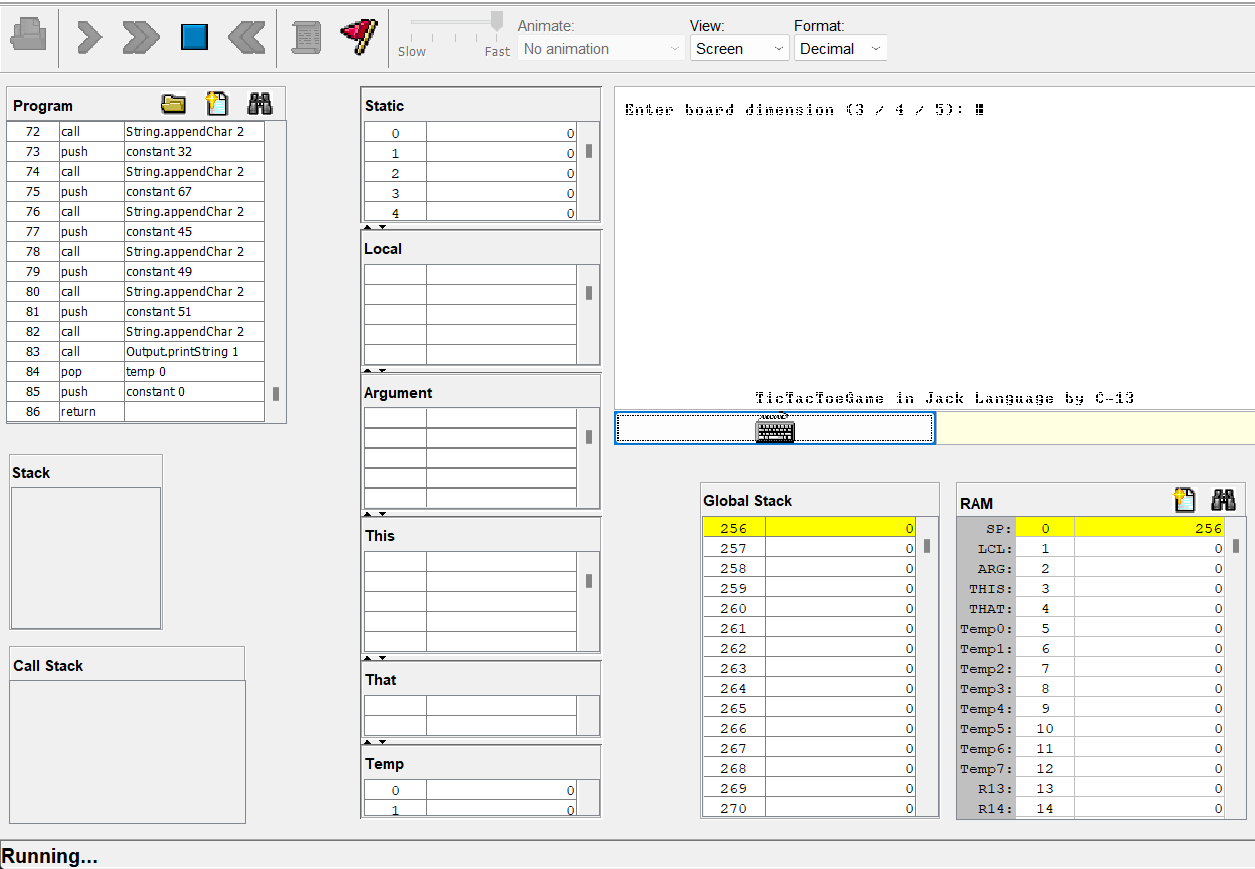
The Tic-Tac-Toe game, also known as Noughts and Crosses, is a classic two-player game where each player alternates placing their symbol, either a naught (O) or a cross (X), on a 3x3 grid or a larger board depending on the game variant. The goal is to be the first to align three of their symbols horizontally, vertically, or diagonally. Our implementation of Tic-Tac-Toe utilizes the Jack language, designed for the Nand2Tetris project. The game consists of several classes, each responsible for different aspects: the Piece class manages the game pieces, InputUtil handles user input and message display, the Board class manages the game board and logic, and the TicTacToeGame class serves as the main controller managing game flow and player interactions. The Main class initializes and starts the game.

The game begins with the Main class, which calls the init method of the TicTacToeGame class to set up game components, followed by the startGame method to display game credits and prompt players to choose the board dimension. Players take turns making moves by entering their chosen cell number, processed by the play method. This method updates the board and checks for game status, such as continuation, win, draw, or quit. Players can undo their last move using u or U, reverting the move and updating the game state. When a player aligns their pieces to form a winning line, the evaluateBoard method identifies the win, and the drawWin method highlights the winning line. If all cells are filled without a win, the game is declared a draw, and the game loop ends. After a game ends, players can choose to restart or quit. If they restart, the game resets and reinitializes the board for a new game.

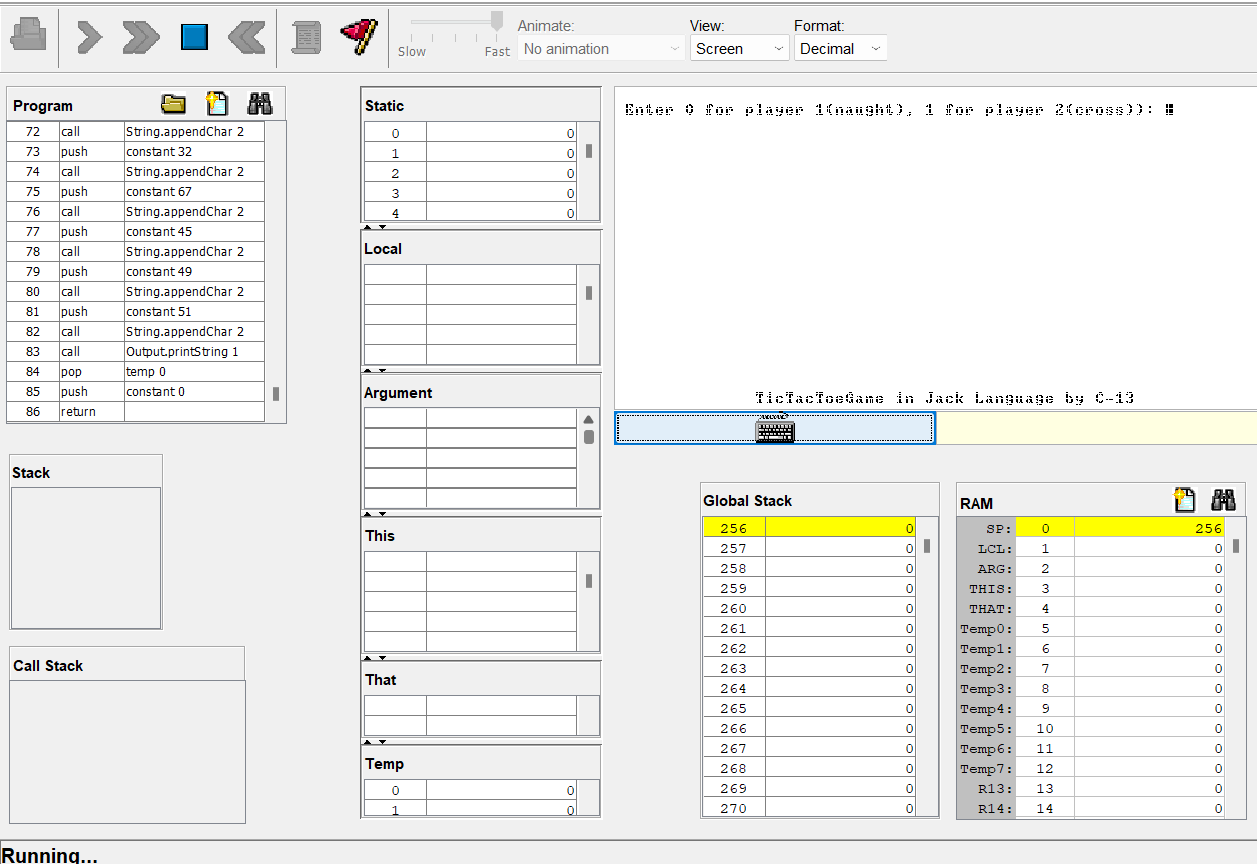
This implementation ensures a comprehensive, engaging, and smooth gameplay experience, from initialization through game play and ending conditions, whether a win, draw, or game restart.

**Screen Results of TicTacToe Game :**

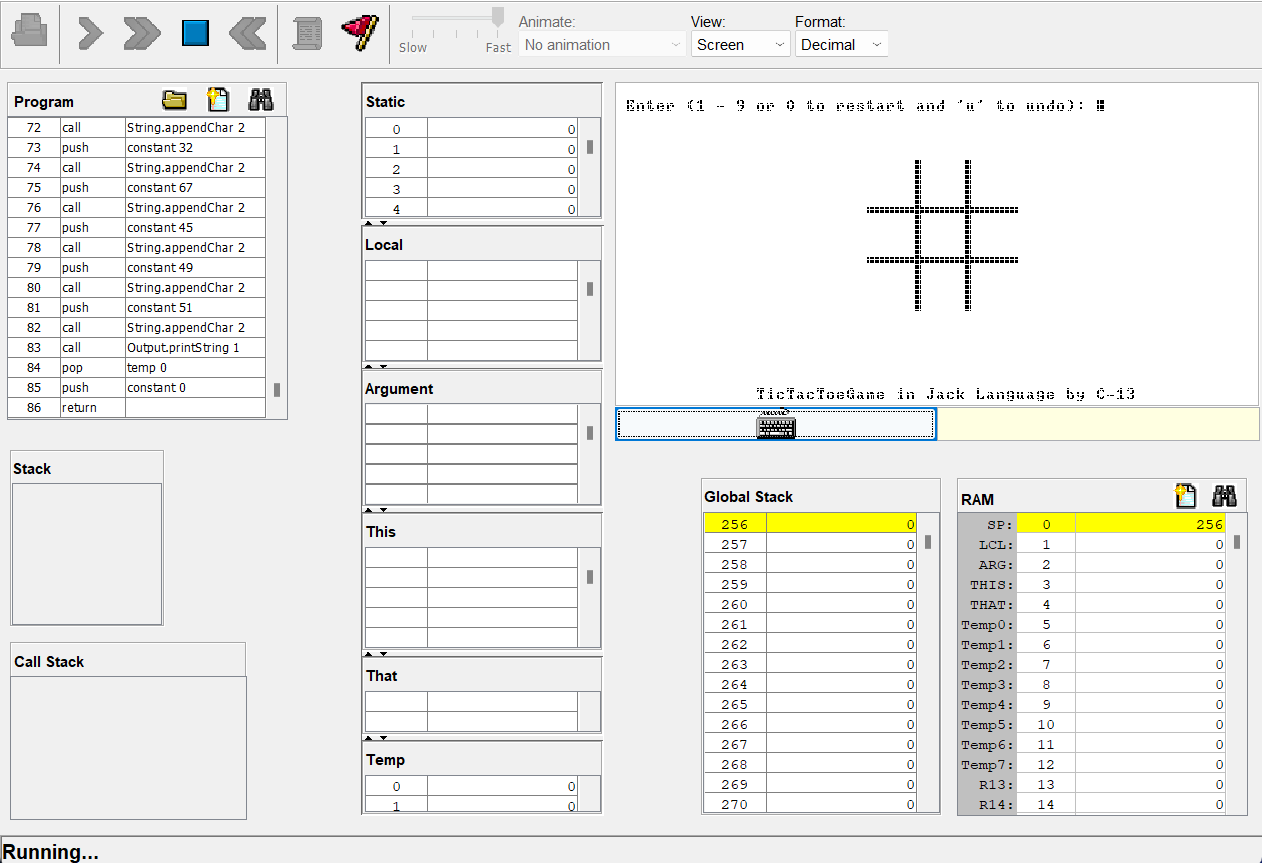
Choose the board Dimension 3 - 3x3, 4 - 4x4, 5 - 5x5

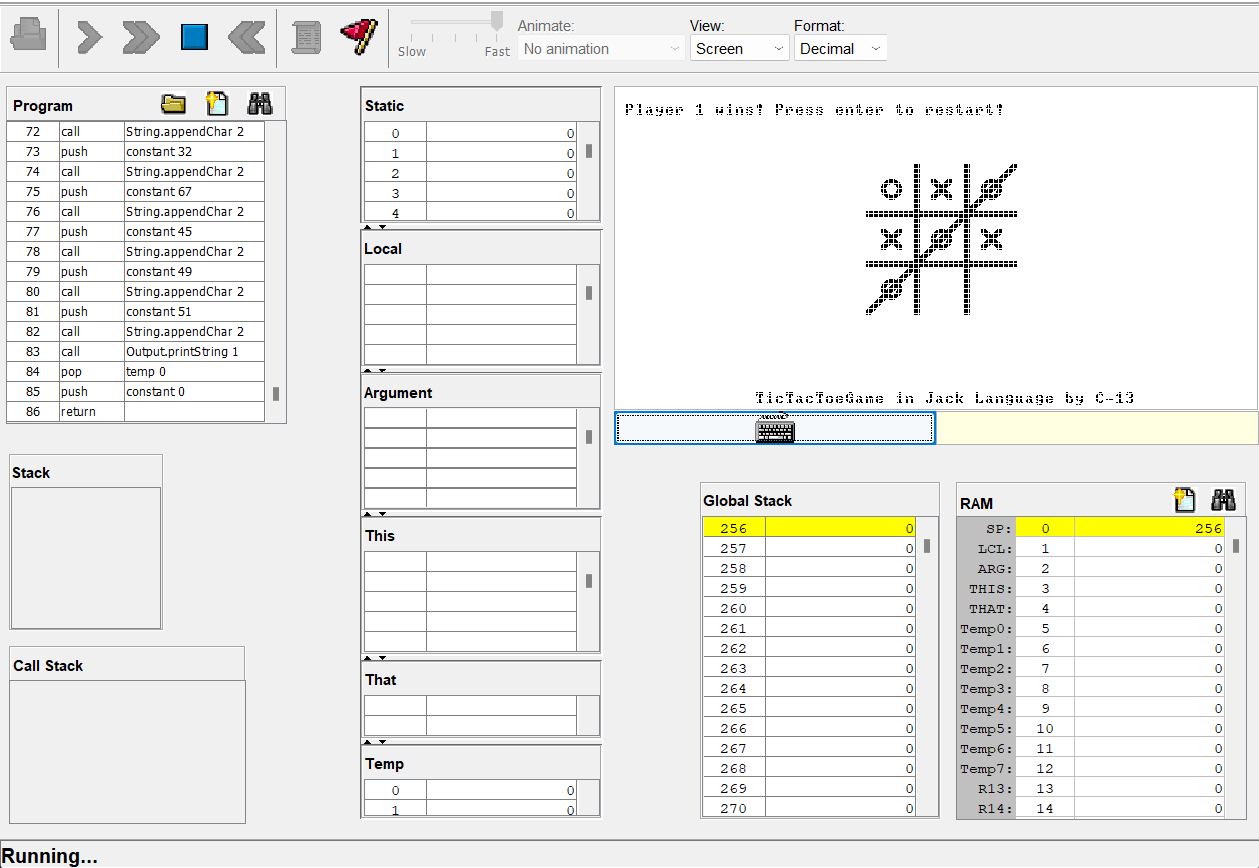


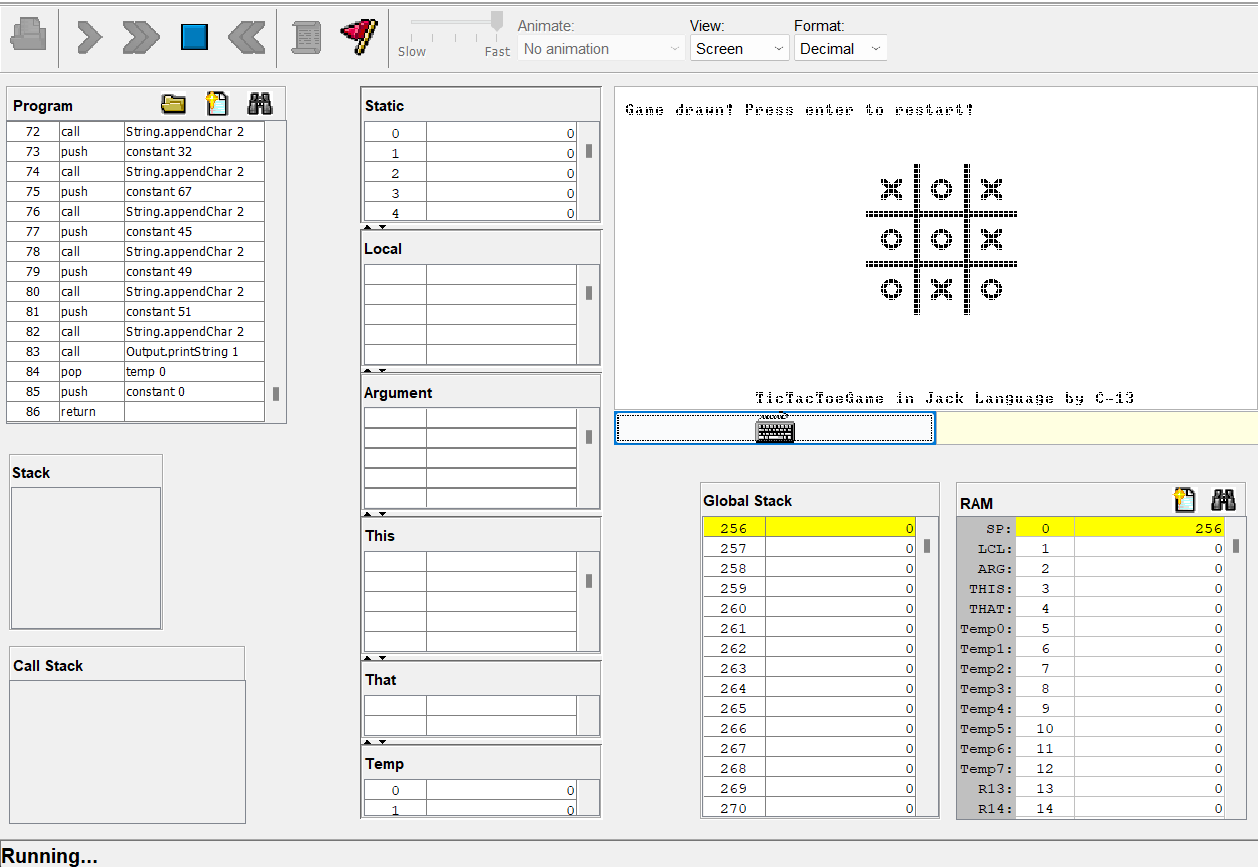
Choose the players and pieces (0 for player 1(naught piece), and 1 for player 2(cross piece))



3x3 board takes input from 1 to 9 and 0 to restart and ‘u’ for undo

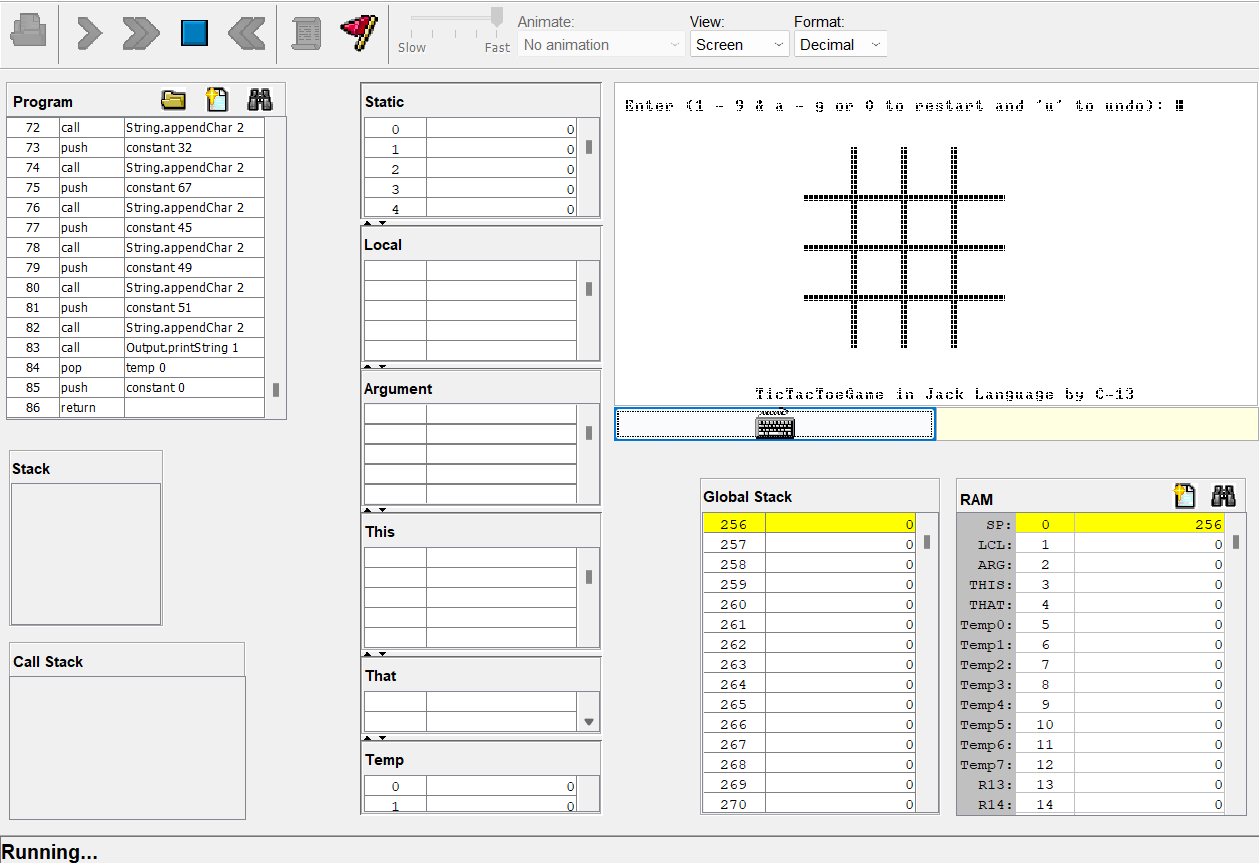


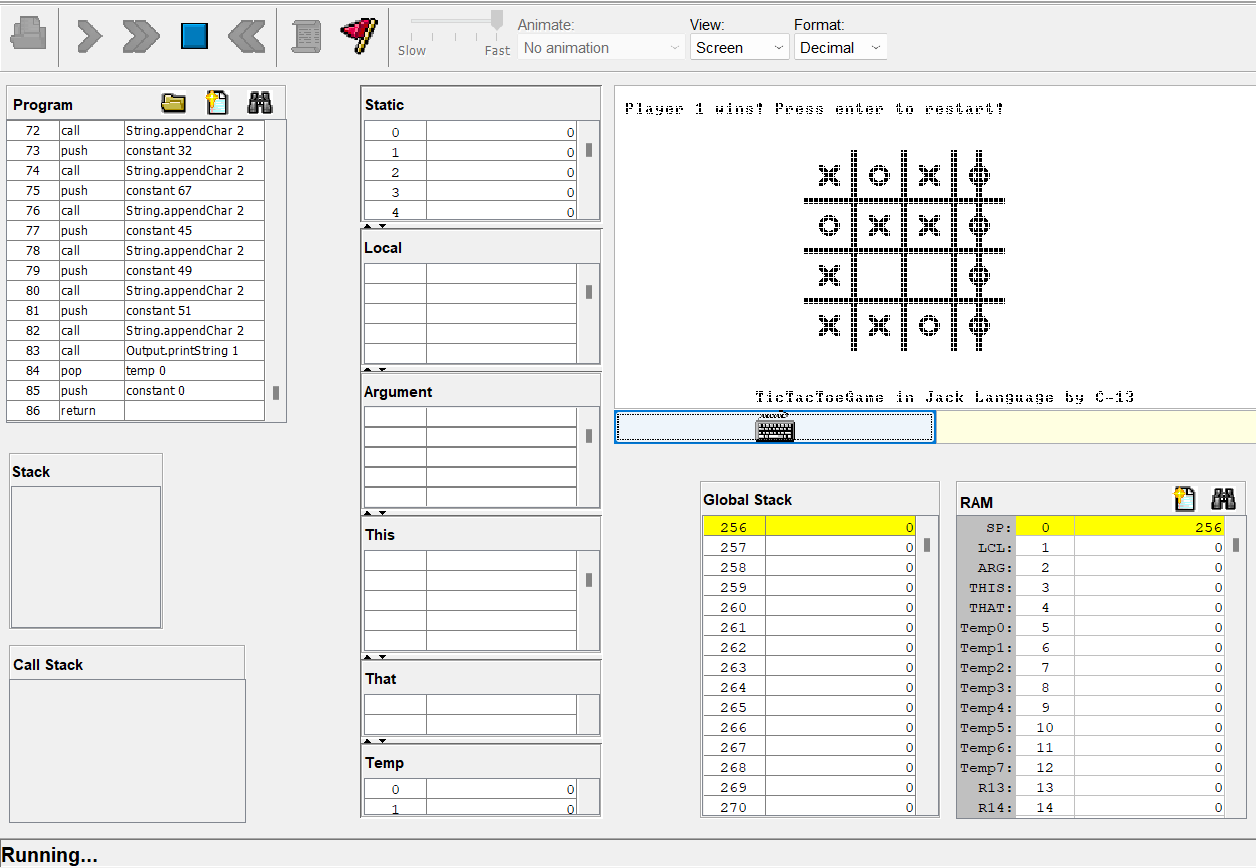


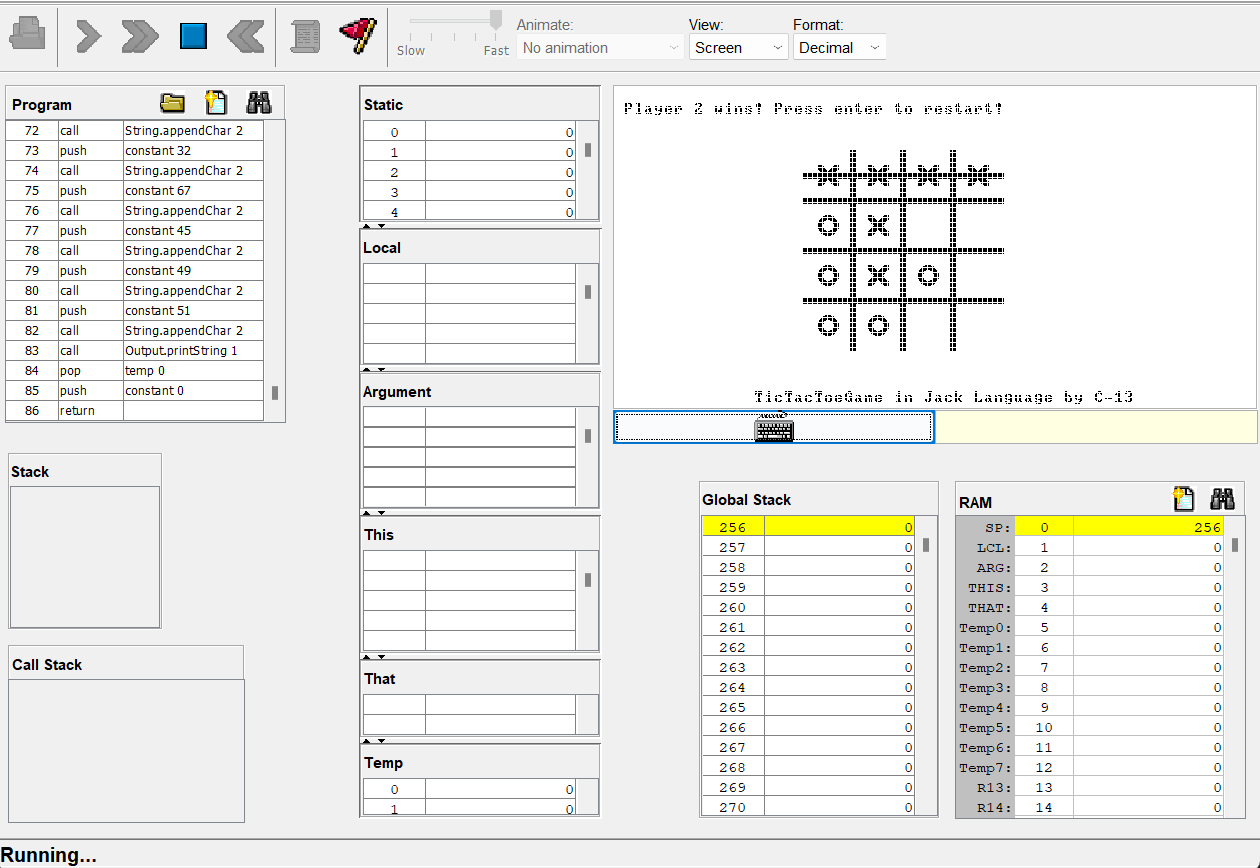


Players need close the application and reopen it to change the board dimension and play the game.

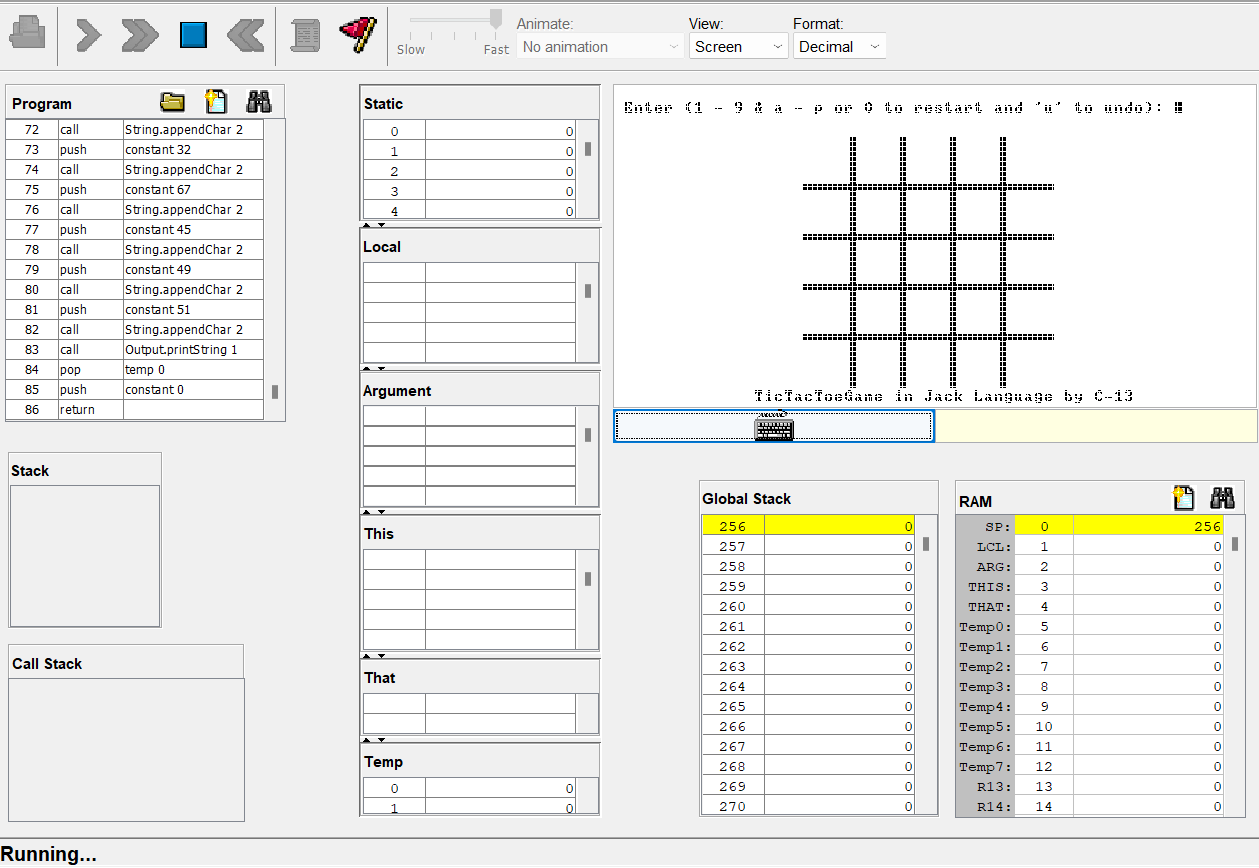
4x4 board takes 16 inputs, the input are 1 to 9 and ‘a’ to ‘g’ and 0 to restart and ‘u’ for undo

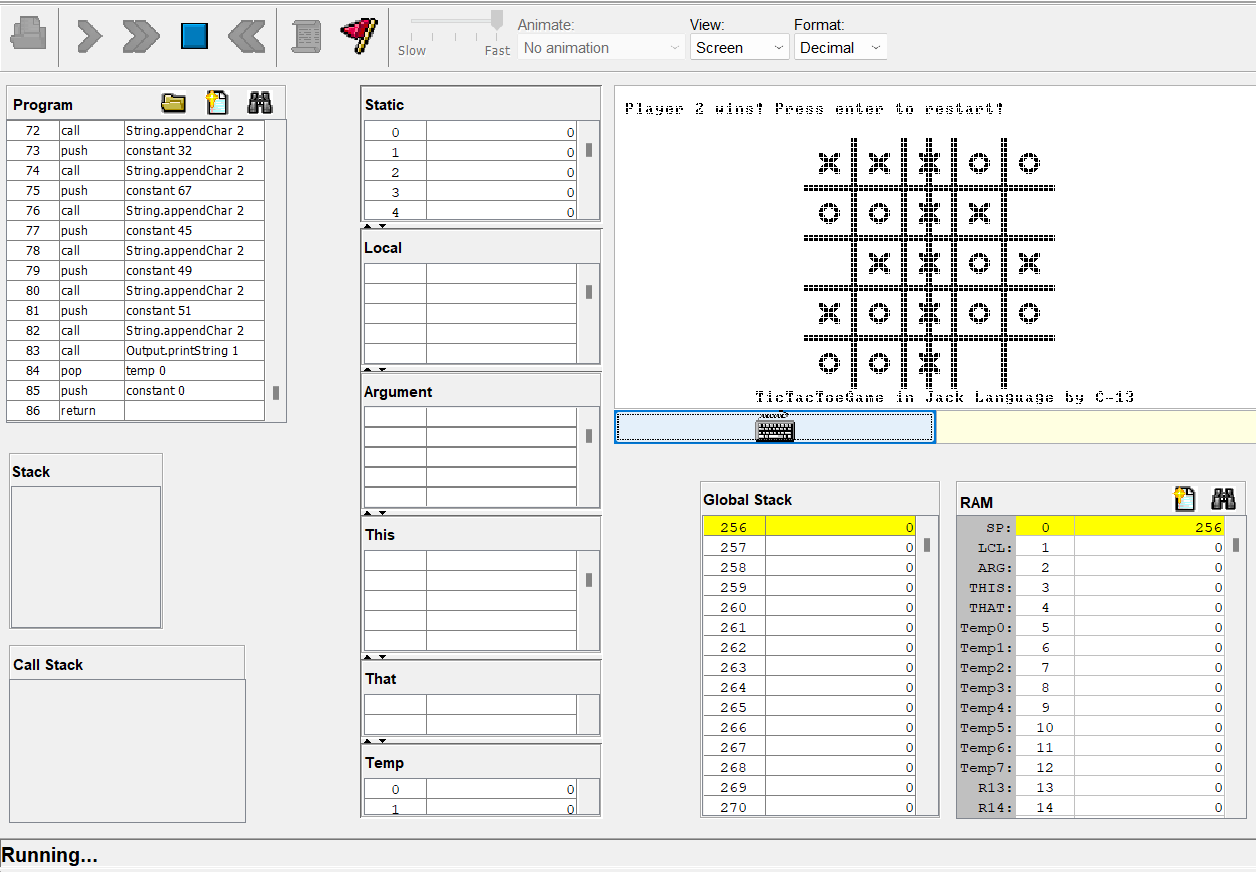


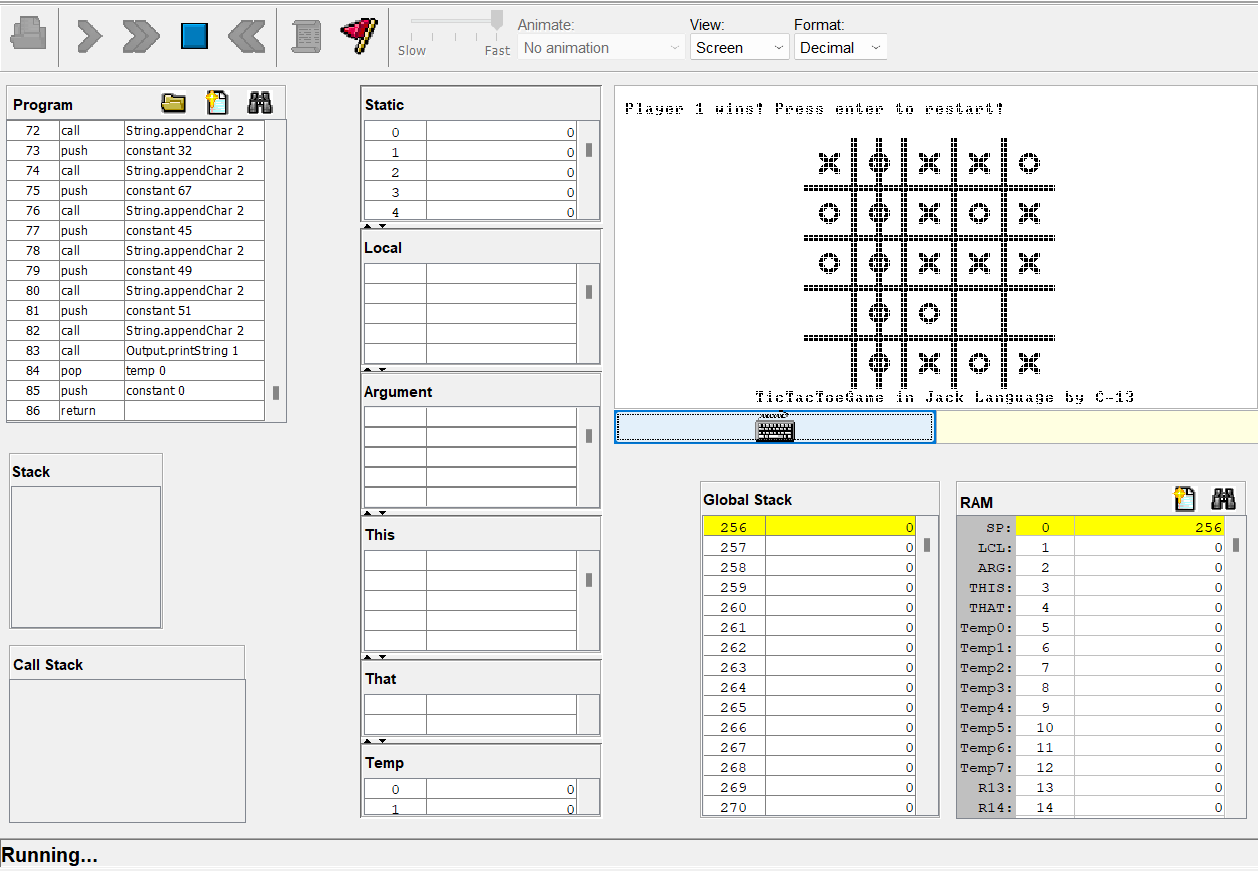


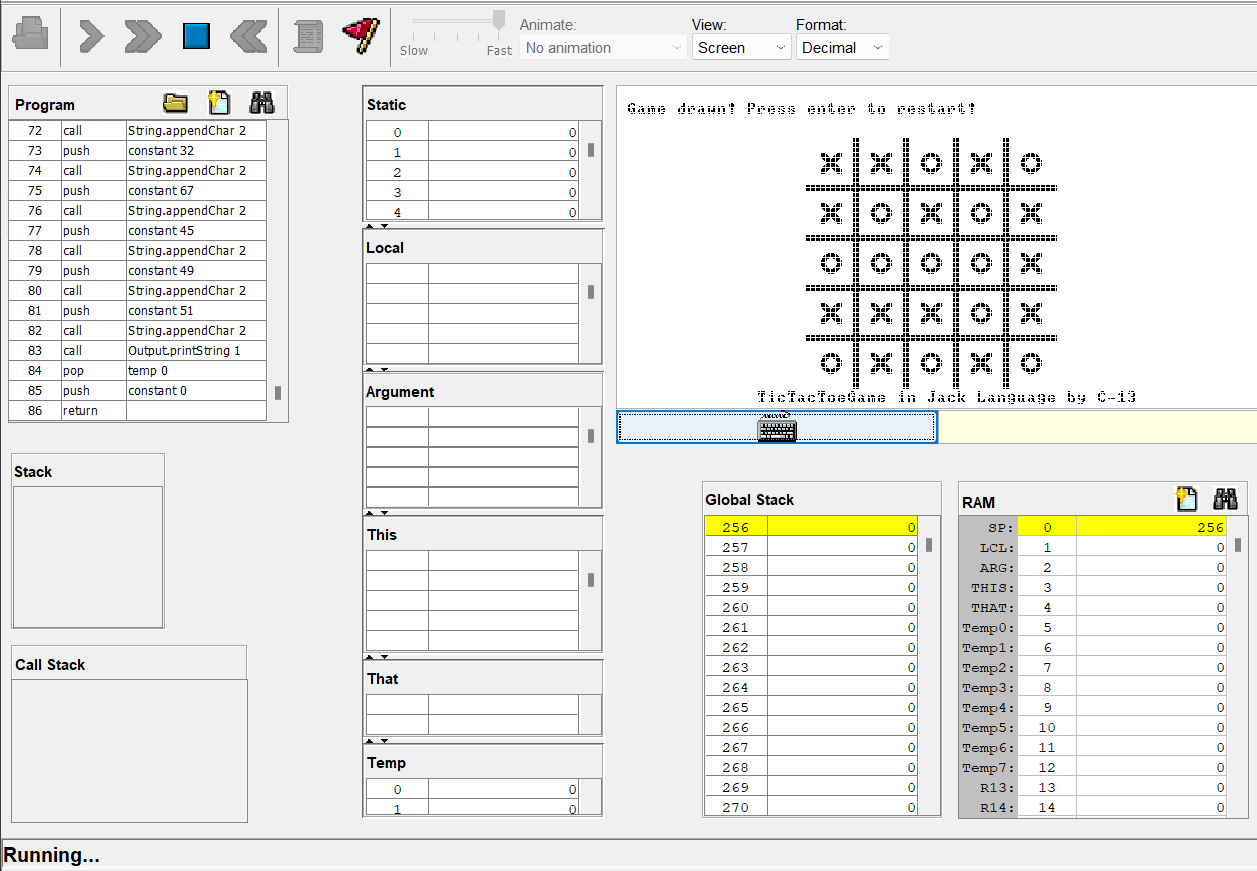


5x5 board takes 25 inputs, the input are 1 to 9 and ‘a’ to ‘p’ and 0 to restart and ‘u’ for undo









1. **RESULTS AND DISCUSSION**

The game performs as expected, delivering an interactive and enjoyable experience for two players. During testing, key features such as move placement, turn alternation, win detection, and handling of draw situations were all verified to be working correctly. The undo move feature enhances usability by allowing players to correct mistakes without restarting. The graphical interface, managed by the Piece and Board classes, accurately renders the game board and pieces, providing clear visual feedback for each move. The InputUtil class efficiently handles user inputs, ensuring smooth gameplay across different board sizes, including 3x3, 4x4, and 5x5 grids.

The evaluation revealed several strengths, including the code's clarity and maintainability due to well-organized classes and methods. The `Board` class effectively encapsulates game logic, managing the state, detecting winning conditions, and handling moves efficiently. Using the Jack language within the Nand2Tetris project emphasizes the educational value, reinforcing essential concepts of computer architecture and software development. However, there are limitations, such as the absence of an AI opponent, limiting the game to player vs. player mode. While the undo feature is useful, it could be improved by allowing a multi-step undo rather than just a single move. Future improvements could involve adding an AI opponent for single-player mode, enhancing the user interface for a more modern appearance, and expanding the game to support larger grids or additional variations. Overall, this project not only serves as a practical exercise in programming but also provides insights into the challenges of game development within an educational context.

1. **CONCLUSION**

The implementation of the Tic-Tac-Toe game using the Jack language within the Nand2Tetris framework has been a successful endeavor, demonstrating the effective application of object-oriented programming principles in an educational setting. The project achieved its objectives by providing a functional, interactive game that incorporates essential features such as turn-based play, win and draw detection, and an undo feature. This project not only enhanced understanding of programming concepts but also highlighted the intricacies of game development and user interface design.

Moreover, this project emphasized the educational value of integrating theoretical knowledge with practical application. By working within the constraints of the Nand2Tetris framework, the project reinforced important concepts of computer architecture and software engineering. Despite some limitations, such as the lack of an AI opponent and the potential for further enhancement of the undo feature, the project lays a solid foundation for future improvements. Overall, this project provided valuable insights and practical experience in software development, making it a worthwhile educational tool.

1. **REFERENCE**

1. Nisan, N., & Schocken, S. (2008). The Elements of Computing Systems: Building a Modern Computer from First Principles.

2. Smith, J., & Goodwin, M. (2010). Enhancing Learning in Computer Architecture through Project-Based Learning.

3. Garcia, A., Romero, P., & Gonzalez, M. (2013). The Impact of Game Development Projects on Students' Learning Outcomes in Computer Science.

4. Joseph, Jayson. (2017) Tic Tac Toe game implementation in Jack language for the Hack platform (Nand2Tetris)