README

Identification of the Topic and Group

Topic: Game of STAQS

Group Designation: STAQS_9

Members:

1. **Student Number:** up202108698

   Full Name: Tomás Rebelo da Silva

   Contribution: 100%

   Tasks Performed: All of them

Installation and Execution

Prerequisites

- SICStus Prolog 4.9

Linux and Windows

1. **Install SICStus Prolog 4.9:** Follow the instructions on the SICStus Prolog website to download and install the appropriate version for your Linux distribution.

Instalation

1. **Clone the Git Repository:**

```
git clone https://github.com/tomassilva03/PFL-TP2
```

Execution

1. **Launch SICStus and consult game.pl:**

```
['/path/to/file/game.pl'].
```

2. **Initiate the Game:**

```
play.
```

3. **Select desired game mode, board size and consequent difficulty in SICStus:**

```
Welcome to the Game of STAQS!
Select game type:
1. Human vs Human (H/H)
2. Human vs Computer (H/PC)
3. Computer vs Human (PC/H)
4. Computer vs Computer (PC/PC)
Enter your choice (1-4): 4.
Enter the board size from 4 to 10 (e.g., 5 for a 5x5 board): 5.
```

Don't forget to put '.' at the end of each input.

4. **Play the game.**

Description of the Game

STAQS is a strategic board game where players compete to control the tallest stacks of pieces on a grid. The game is played on a square board, with each cell initially containing a neutral piece. Players take turns placing their pieces on the board or stacking their pieces on top of existing stacks. The objective is to create the tallest stack of pieces by the end of the game.

Rules:

1. **Setup Phase:**
    - Players take turns placing their pieces on any neutral cell on the board.
    - The setup phase continues until all pieces are placed.
2. **Play Phase:**
    - Players take turns moving their stacks.
    - Players can only stack their pieces on top of their own stacks or neutral stacks.
    - Stacks can only be moved orthogonally (horizontally or vertically) to an adjacent cell.
3. **End of the Game:**
    - The game ends when no more valid moves are available for both players.
    - The player with the tallest stack wins the game.
    - In case of a tie in the tallest stack, the player with the second tallest stack wins.

References:

- Official Game Website

Considerations for Game Extensions

Variable-Sized Boards

When extending the game design to support variable-sized boards, the following considerations were taken into account:

1. **Board Initialization:**
    - The game should allow players to choose the board size at the start of the game.
    - The board size can range from 4x4 to 10x10, providing flexibility for different game durations and strategies.
2. **Game Balance:**
    - The rules and mechanics should be adjusted to ensure balanced gameplay across different board sizes.
    - Larger boards may require additional pieces or modified rules to maintain the game's strategic depth.
3. **User Interface:**
    - The user interface should dynamically adjust to display the chosen board size.
    - Input validation should ensure that players enter valid board sizes within the allowed range.
4. **Performance:**
    - The implementation should be optimized to handle larger board sizes without significant performance degradation.
    - Efficient algorithms for move validation and game state evaluation are crucial for maintaining a smooth gameplay experience.

These considerations ensure that the game remains engaging and balanced, regardless of the chosen board size.

Game Logic

Game Configuration Representation

The game configuration is represented by a list of parameters that define the initial setup of the game. This includes the board size, player types, optional rules, and player names. The `initial_state/2` predicate uses this configuration to set up the initial game state.

Internal Game State Representation

The internal game state is represented by a `state/5` structure:

- `Board`: A list of lists representing the board, where each cell contains a piece or stack in the format `Player-Count`.
- `CurrentPlayer`: The player whose turn it is to move.
- `Pieces`: A list of remaining pieces for each player.
- `Phase`: The current phase of the game (`setup` or `play`).
- `BoardSize`: The size of the board.

Move Representation

Moves are represented by structures indicating the type of move and the coordinates involved:

- `place(Y, X)`: Place a piece at the specified coordinates during the setup phase.
- `stack(Y1, X1, Y2, X2)`: Move a stack from one cell to an adjacent cell during the play phase.
- `skip`: Skip the turn if no valid moves are available.

The `move/3` predicate applies the move to the game state and returns the new state.

User Interaction

The game menu system allows players to select the game type, board size, and difficulty level. Interaction with the user is performed through prompts and input validation to ensure valid moves are entered. The `get_player_move/4` predicate handles user input and validates moves.

Conclusions

Limitations

One of the main limitations of the current implementation is the lack of strategic depth for the minimax algorithm during the setup phase. While the greedy algorithm has been implemented to add strategic decision-making during this phase, the minimax AI still relies on random move selection, regardless of the chosen difficulty level. This reduces the effectiveness of the AI's strategy at the start of the game.

Another limitation is the inability to perform comprehensive input validation for some inputs. While player moves during the game are validated, inputs such as game type, board size, and difficulty level are not fully validated, which may lead to unexpected behavior or errors.

Possible Improvements

1. **Enhanced AI for Setup Phase:**
    - Develop more sophisticated algorithms for the setup phase to introduce strategic depth and make the game more challenging from the beginning.
2. **Additional Game Modes:**
    - Introduce new game modes with different rules or objectives to provide variety and cater to different player preferences.
3. **Improved User Interface:**
    - Enhance the user interface to provide a more intuitive and visually appealing experience, including better input validation and error handling.
4. **Performance Optimization:**
    - Further optimize the game logic to handle larger board sizes and more complex game states efficiently.

Future Developments Roadmap

1. **AI Enhancements:**

- Implement advanced AI strategies for both the setup and play phases, including machine learning techniques to adapt to player behavior.

Conclusions about the Work Carried Out

The development of the STAQS game in Prolog has been a valuable learning experience, providing insights into game design, AI algorithms, and user interaction. The project successfully implemented a functional game with variable-sized boards and different difficulty levels. However, there are areas for improvement, particularly in enhancing the AI's strategic capabilities during the setup phase and optimizing performance for larger boards. Additionally, improving input validation for all user inputs is necessary to ensure a robust and error-free gaming experience. Overall, the project demonstrates the potential of Prolog for developing complex logic-based games and sets the foundation for future enhancements and extensions.

Minimax Algorithm Strategy

Overview

The minimax algorithm is employed to select the most advantageous move for the AI by evaluating all potential game states up to a specified depth. The algorithm alternates between maximizing the AI's score and minimizing the opponent's advantage, ensuring the AI makes strategic decisions.

Implementation Details

1. **Dynamic Depth Adjustment**:
    - The depth of the minimax search adapts to the board size to balance computational cost and strategic complexity:
        - **Board Size ≤ 4**: Depth = 6
        - **Board Size = 5**: Depth = 4
        - **Board Size = 6**: Depth = 3
        - **Board Size ≥ 7**: Depth = 2
2. **Move Simulation**:
    - The algorithm simulates all valid moves for both the AI and its opponent.
    - Each move generates a new game state, which is recursively evaluated.
3. **Evaluation Function**:
    - During the **setup phase**, the value of a move is based on:
        - Proximity to friendly stacks to encourage clustering.
        - Proximity to opponent stacks to disrupt their strategy.
        - Proximity to the center of the board for better control.
    - During the **play phase**, the value of a move is determined by:
        - The height of the resulting stack.
        - Strategic proximity to friendly and opponent stacks.
    - **Formula for setup phase**:

```
Value = (FriendlyProximity × 2) + OpponentProximity +
(CenterScore × 3)
```

- **Formula for play phase**:

```
   Value = (StackHeight × 3) + FriendlyProximity +
OpponentProximity
```

4. **Maximizing and Minimizing**:
   - The algorithm alternates between choosing the move with the highest score for the AI and the lowest score for the opponent.
5. **Base Case**:
   - When the specified depth is reached or no valid moves are available, the evaluation function assigns a static score to the game state.
6. **Best Move Selection**:
   - The algorithm evaluates all moves and selects the one with the highest score for the AI.
   - In case of multiple moves with the same score, a random selection is made among them.

Key Features:

- The implementation integrates move simulation, evaluation, and strategic depth adjustment, making the AI both efficient and strategic.
- The AI adapts dynamically to varying board sizes and phases of the game, ensuring optimal gameplay across scenarios.

Tests

Overview

The STAQS game includes a comprehensive suite of tests to ensure the correctness and robustness of the game logic. The tests are implemented using the Prolog `plunit` library and cover various aspects of the game, including game state transitions, move validation, and AI behavior.

Running the Tests

To run the tests, follow these steps:

1. **Launch SICStus Prolog and consult the `game.pl` file:**

```
['/path/to/file/game.pl'].
```

2. **Run the tests:**

```
run_tests.
```

Tests

Overview

The STAQS game includes a comprehensive suite of tests implemented using the Prolog `plunit` library. These tests ensure the correctness and robustness of the game's logic, including game state transitions, move validation, and AI behavior.

Running the Tests

To run the tests, follow these steps:

1. **Launch SICStus Prolog and consult the `game.pl` file:**

```
['path/to/game.pl'].
```

2. **Run the tests:**

```
run_tests.
```

Test Categories

1. **Game Mechanics Tests:** This section focuses on validating the core game mechanics, including transitions between phases and game-ending conditions. Tests ensure that the game behaves correctly under various scenarios, such as determining the winner, handling draws, and transitioning from the setup phase to the play phase.
2. **Player Actions Tests:** These tests verify the validity of player actions during the game. They cover scenarios like skipping turns, placing pieces, stacking moves, and handling invalid moves (e.g., placing a piece on an occupied cell or stacking onto non-adjacent cells). This section ensures that players and the game logic follow the rules correctly.
3. **AI Behavior Tests:** This section tests the behavior of different AI strategies, including random, greedy, and minimax algorithms. It validates that the AI makes legal and strategic moves according to its selected difficulty level and gameplay phase. Tests also evaluate the AI's ability to maximize short-term or long-term advantages based on the chosen strategy.

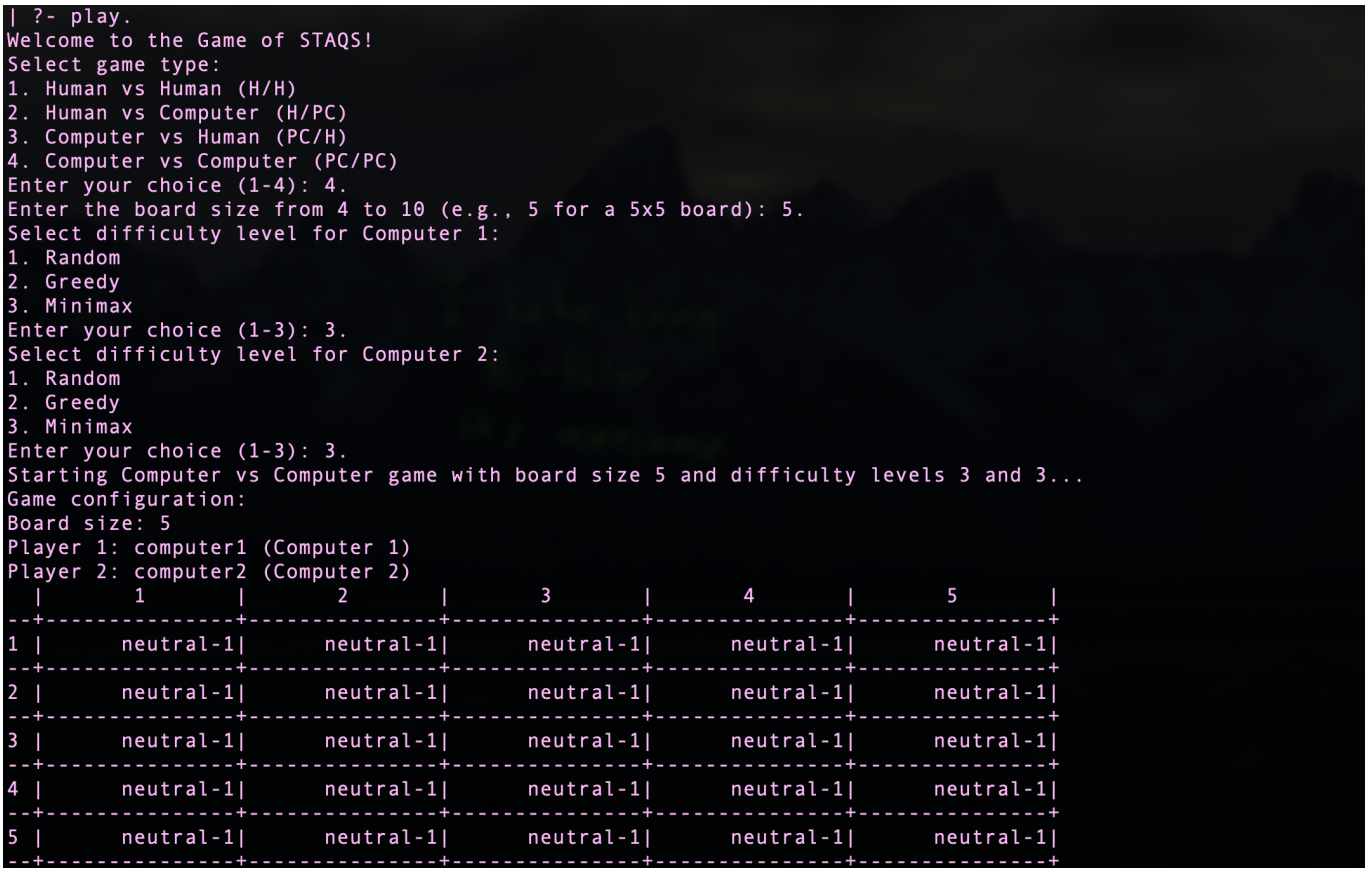
Notes

- The tests are structured to cover all critical aspects of the game's logic and functionality.

- By running these tests, you can ensure the game's reliability and identify potential issues in edge cases or complex scenarios.
- The comprehensive test suite provides confidence in the game's implementation, especially for AI behavior and rule enforcement.

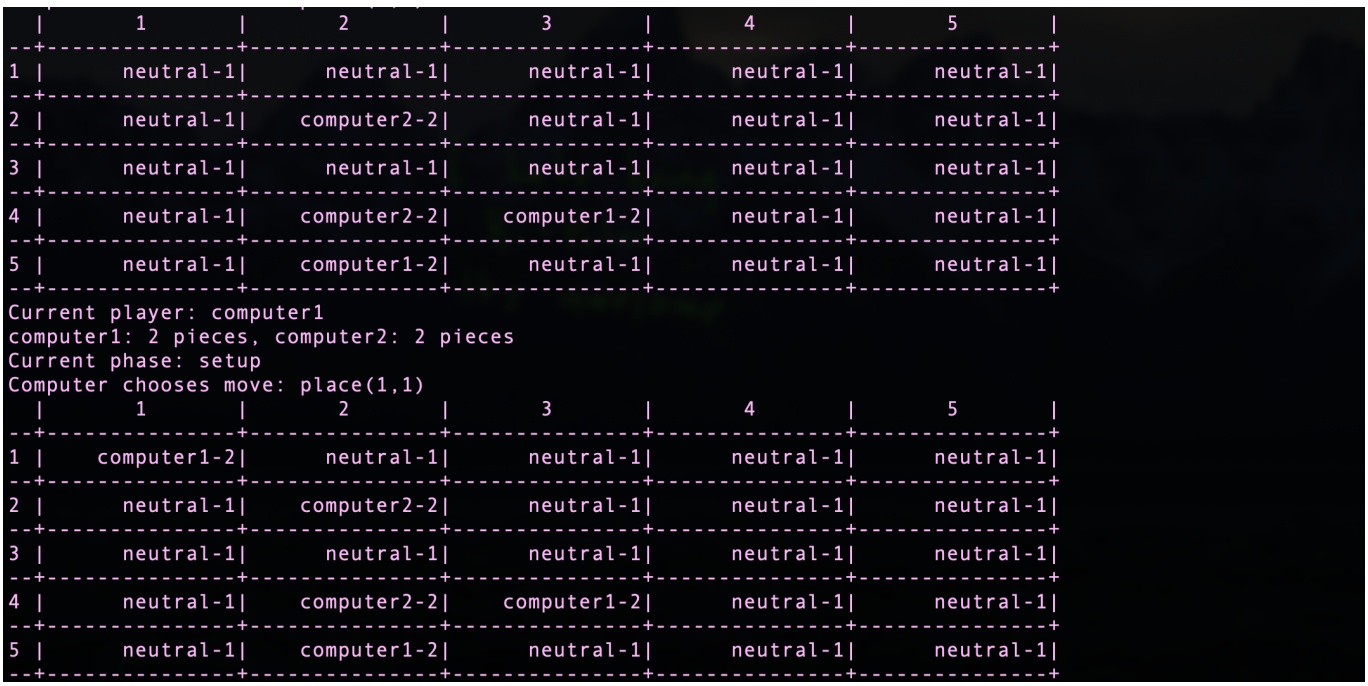
Images

Initial Menu and Board Setup



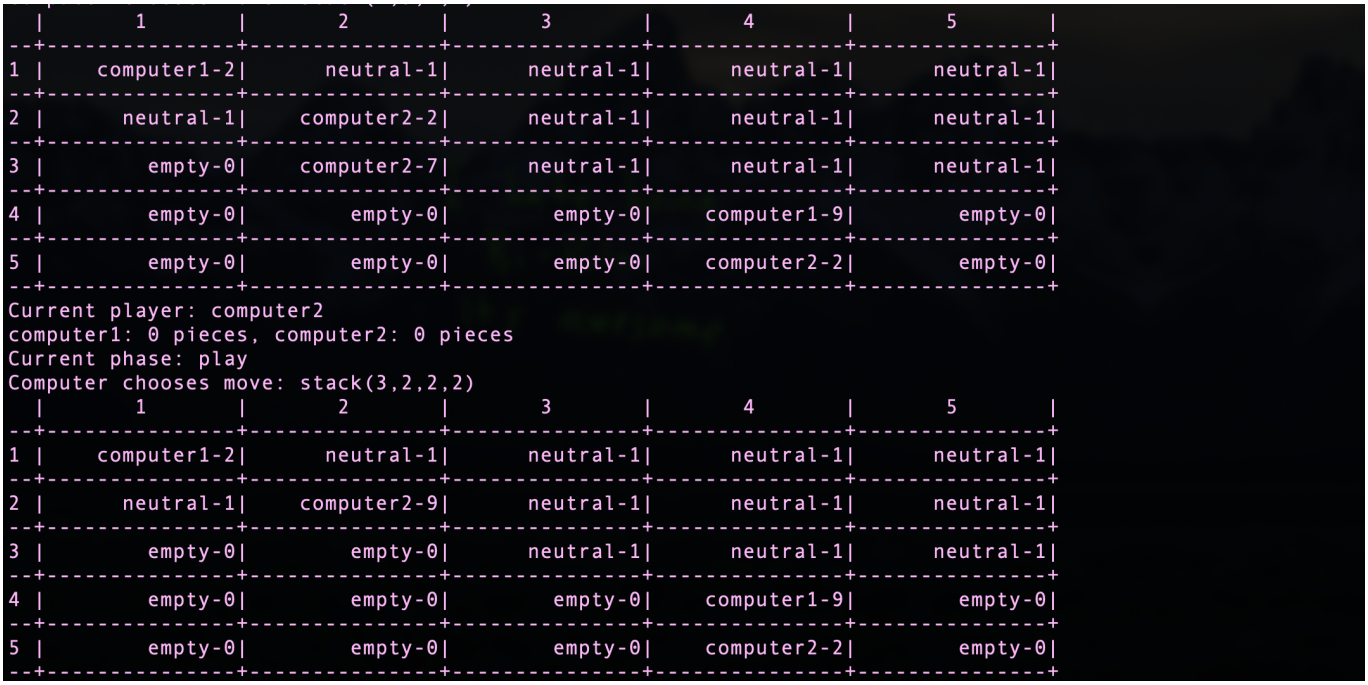
```
| ?- play.
Welcome to the Game of STAQS!
Select game type:
1. Human vs Human (H/H)
2. Human vs Computer (H/PC)
3. Computer vs Human (PC/H)
4. Computer vs Computer (PC/PC)
Enter your choice (1-4): 4.
Enter the board size from 4 to 10 (e.g., 5 for a 5x5 board): 5.
Select difficulty level for Computer 1:
1. Random
2. Greedy
3. Minimax
Enter your choice (1-3): 3.
Select difficulty level for Computer 2:
1. Random
2. Greedy
3. Minimax
Enter your choice (1-3): 3.
Starting Computer vs Computer game with board size 5 and difficulty levels 3 and 3...
Game configuration:
Board size: 5
Player 1: computer1 (Computer 1)
Player 2: computer2 (Computer 2)
  |       1       |       2       |       3       |       4       |       5       |
--+---------------+---------------+---------------+---------------+---------------+
1 |      neutral-1|      neutral-1|      neutral-1|      neutral-1|      neutral-1|
--+---------------+---------------+---------------+---------------+---------------+
2 |      neutral-1|      neutral-1|      neutral-1|      neutral-1|      neutral-1|
--+---------------+---------------+---------------+---------------+---------------+
3 |      neutral-1|      neutral-1|      neutral-1|      neutral-1|      neutral-1|
--+---------------+---------------+---------------+---------------+---------------+
4 |      neutral-1|      neutral-1|      neutral-1|      neutral-1|      neutral-1|
--+---------------+---------------+---------------+---------------+---------------+
5 |      neutral-1|      neutral-1|      neutral-1|      neutral-1|      neutral-1|
--+---------------+---------------+---------------+---------------+---------------+
```

Move in the Setup Phase



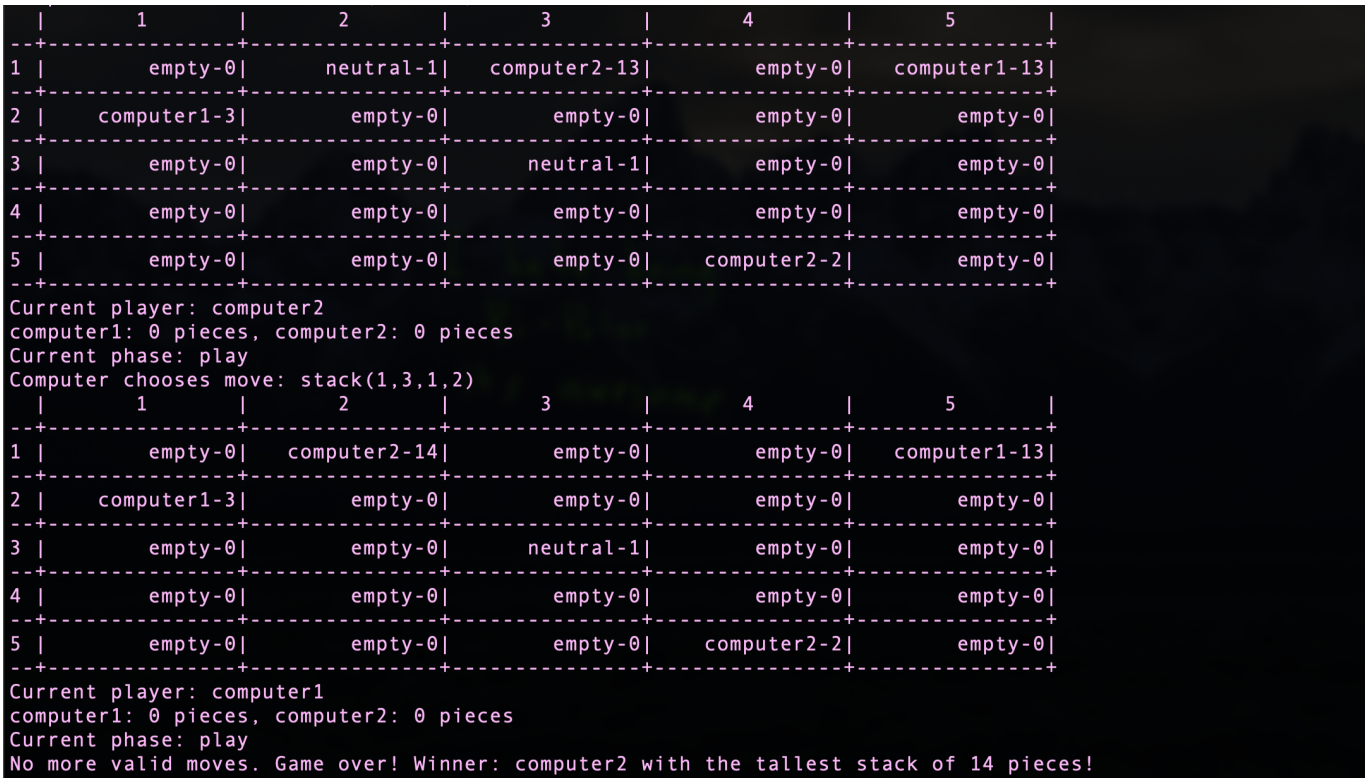
```
  |       1       |       2       |       3       |       4       |       5       |
--+---------------+---------------+---------------+---------------+---------------+
1 |      neutral-1|      neutral-1|      neutral-1|      neutral-1|      neutral-1|
--+---------------+---------------+---------------+---------------+---------------+
2 |      neutral-1|    computer2-2|      neutral-1|      neutral-1|      neutral-1|
--+---------------+---------------+---------------+---------------+---------------+
3 |      neutral-1|      neutral-1|      neutral-1|      neutral-1|      neutral-1|
--+---------------+---------------+---------------+---------------+---------------+
4 |      neutral-1|    computer2-2|    computer1-2|      neutral-1|      neutral-1|
--+---------------+---------------+---------------+---------------+---------------+
5 |      neutral-1|    computer1-2|      neutral-1|      neutral-1|      neutral-1|
--+---------------+---------------+---------------+---------------+---------------+
Current player: computer1
computer1: 2 pieces, computer2: 2 pieces
Current phase: setup
Computer chooses move: place(1,1)
  |       1       |       2       |       3       |       4       |       5       |
--+---------------+---------------+---------------+---------------+---------------+
1 |    computer1-2|      neutral-1|      neutral-1|      neutral-1|      neutral-1|
--+---------------+---------------+---------------+---------------+---------------+
2 |      neutral-1|    computer2-2|      neutral-1|      neutral-1|      neutral-1|
--+---------------+---------------+---------------+---------------+---------------+
3 |      neutral-1|      neutral-1|      neutral-1|      neutral-1|      neutral-1|
--+---------------+---------------+---------------+---------------+---------------+
4 |      neutral-1|    computer2-2|    computer1-2|      neutral-1|      neutral-1|
--+---------------+---------------+---------------+---------------+---------------+
5 |      neutral-1|    computer1-2|      neutral-1|      neutral-1|      neutral-1|
--+---------------+---------------+---------------+---------------+---------------+
```

Move in the Play Phase



```
  |       1       |       2       |       3       |       4       |       5       |
--+---------------+---------------+---------------+---------------+---------------+
1 |    computer1-2|      neutral-1|      neutral-1|      neutral-1|      neutral-1|
--+---------------+---------------+---------------+---------------+---------------+
2 |      neutral-1|    computer2-2|      neutral-1|      neutral-1|      neutral-1|
--+---------------+---------------+---------------+---------------+---------------+
3 |        empty-0|    computer2-7|      neutral-1|      neutral-1|      neutral-1|
--+---------------+---------------+---------------+---------------+---------------+
4 |        empty-0|        empty-0|        empty-0|    computer1-9|        empty-0|
--+---------------+---------------+---------------+---------------+---------------+
5 |        empty-0|        empty-0|        empty-0|    computer2-2|        empty-0|
--+---------------+---------------+---------------+---------------+---------------+
Current player: computer2
computer1: 0 pieces, computer2: 0 pieces
Current phase: play
Computer chooses move: stack(3,2,2,2)
  |       1       |       2       |       3       |       4       |       5       |
--+---------------+---------------+---------------+---------------+---------------+
1 |    computer1-2|      neutral-1|      neutral-1|      neutral-1|      neutral-1|
--+---------------+---------------+---------------+---------------+---------------+
2 |      neutral-1|    computer2-9|      neutral-1|      neutral-1|      neutral-1|
--+---------------+---------------+---------------+---------------+---------------+
3 |        empty-0|        empty-0|      neutral-1|      neutral-1|      neutral-1|
--+---------------+---------------+---------------+---------------+---------------+
4 |        empty-0|        empty-0|        empty-0|    computer1-9|        empty-0|
--+---------------+---------------+---------------+---------------+---------------+
5 |        empty-0|        empty-0|        empty-0|    computer2-2|        empty-0|
--+---------------+---------------+---------------+---------------+---------------+
```

Final Move and Final Board



```
  |       1       |       2       |       3       |       4       |       5       |
--+---------------+---------------+---------------+---------------+---------------+
1 |        empty-0|      neutral-1|   computer2-13|        empty-0|   computer1-13|
--+---------------+---------------+---------------+---------------+---------------+
2 |    computer1-3|        empty-0|        empty-0|        empty-0|        empty-0|
--+---------------+---------------+---------------+---------------+---------------+
3 |        empty-0|        empty-0|      neutral-1|        empty-0|        empty-0|
--+---------------+---------------+---------------+---------------+---------------+
4 |        empty-0|        empty-0|        empty-0|        empty-0|        empty-0|
--+---------------+---------------+---------------+---------------+---------------+
5 |        empty-0|        empty-0|        empty-0|    computer2-2|        empty-0|
--+---------------+---------------+---------------+---------------+---------------+
Current player: computer2
computer1: 0 pieces, computer2: 0 pieces
Current phase: play
Computer chooses move: stack(1,3,1,2)
  |       1       |       2       |       3       |       4       |       5       |
--+---------------+---------------+---------------+---------------+---------------+
1 |        empty-0|   computer2-14|        empty-0|        empty-0|   computer1-13|
--+---------------+---------------+---------------+---------------+---------------+
2 |    computer1-3|        empty-0|        empty-0|        empty-0|        empty-0|
--+---------------+---------------+---------------+---------------+---------------+
3 |        empty-0|        empty-0|      neutral-1|        empty-0|        empty-0|
--+---------------+---------------+---------------+---------------+---------------+
4 |        empty-0|        empty-0|        empty-0|        empty-0|        empty-0|
--+---------------+---------------+---------------+---------------+---------------+
5 |        empty-0|        empty-0|        empty-0|    computer2-2|        empty-0|
--+---------------+---------------+---------------+---------------+---------------+
Current player: computer1
computer1: 0 pieces, computer2: 0 pieces
Current phase: play
No more valid moves. Game over! Winner: computer2 with the tallest stack of 14 pieces!
```