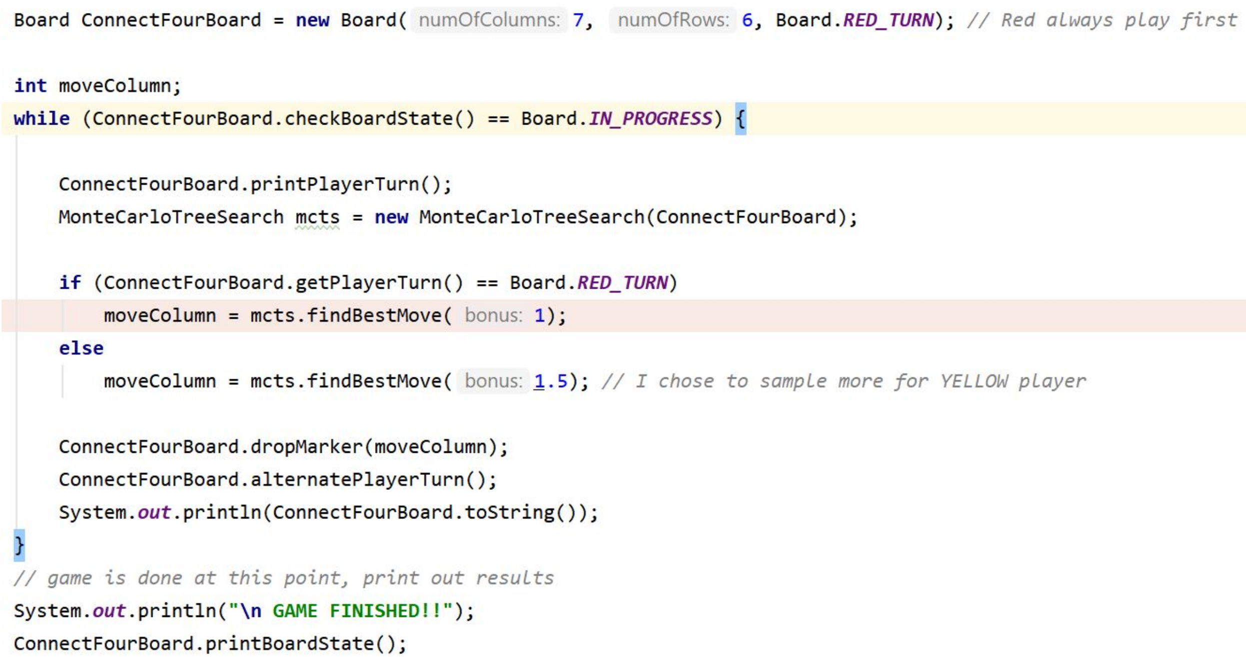
CS 461 - Program 2 Report
Binh Nguyen - 16168354

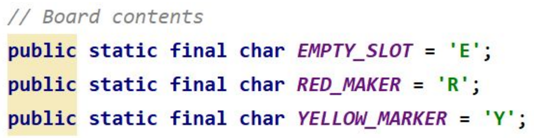
Connect Four Problem with Monte Carlo Tree Search

<u>MAIN CLASSES:</u>

- ***Main.java***
  - Keep playing the game until the game is done.
  - Using monte carlo search method to find the next move for each opponent (in our case RED & YELLOW AI players)
  - 

```
Board ConnectFourBoard = new Board( numOfColumns: 7,  numOfRows: 6, Board.RED_TURN); // Red always play first

int moveColumn;
while (ConnectFourBoard.checkBoardState() == Board.IN_PROGRESS) {

    ConnectFourBoard.printPlayerTurn();
    MonteCarloTreeSearch mcts = new MonteCarloTreeSearch(ConnectFourBoard);

    if (ConnectFourBoard.getPlayerTurn() == Board.RED_TURN)
        moveColumn = mcts.findBestMove( bonus: 1);
    else
        moveColumn = mcts.findBestMove( bonus: 1.5); // I chose to sample more for YELLOW player

    ConnectFourBoard.dropMarker(moveColumn);
    ConnectFourBoard.alternatePlayerTurn();
    System.out.println(ConnectFourBoard.toString());
}
// game is done at this point, print out results
System.out.println("\n GAME FINISHED!!");
ConnectFourBoard.printBoardState();
```

  - Player with <u>50% more searches is YELLOW</u>.
- ***Board.java:***
  - Modeling a connect four board. Using a 2D char array to store content:

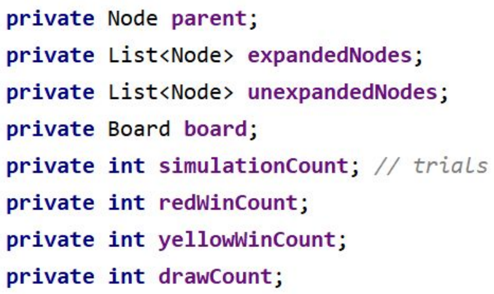


```
// Board contents
public static final char EMPTY_SLOT = 'E';
public static final char RED_MAKER = 'R';
public static final char YELLOW_MARKER = 'Y';
```

  - Game states (win/loss/draw) and player turns are also stored.
  - Has functions to print out player turn, board state, board layout.
  - Has functions to make a play, alternate player turn, and check for a winner.

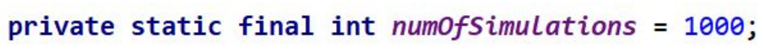
CS 461 - Program 2 Report
Binh Nguyen - 16168354

- ***Node.java***:
    - A search node in Monte Carlo Tree. Core component.
    - Keeping track of game play statistics, children & parent nodes, current board state:



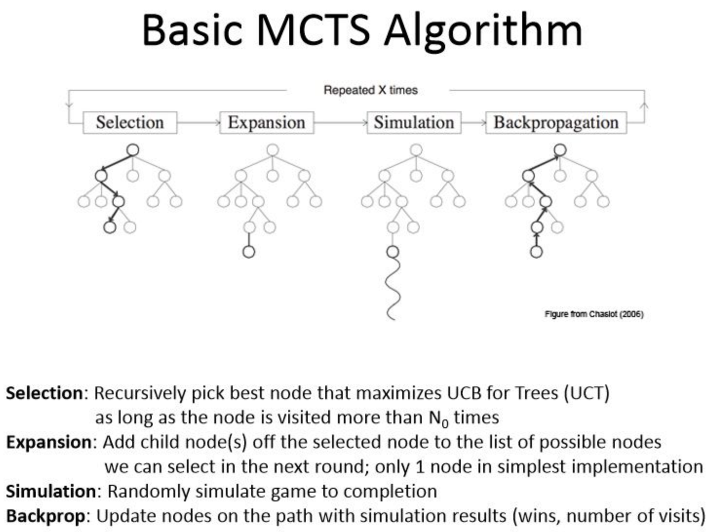
```
private Node parent;
private List<Node> expandedNodes;
private List<Node> unexpandedNodes;
private Board board;
private int simulationCount; // trials
private int redWinCount;
private int yellowWinCount;
private int drawCount;
```

- ***MonteCarloTreeSearch.java***:
    - The main algorithm using Monte Carlo Tree Search method lightweight playout.
    - Run 1000 simulations to find the best move to play



```
private static final int numOfSimulations = 1000;
```

- Has 4 main methods:
    - Select: Traversing down the search tree to select the best move based on UCT value. If node is not fully expanded/leaf node, go to expansion phase.
    - Expand: Randomly pick one of the possible moves from our current node and create a child node according to that move.
    - Simulation: Simulate game with each player making random moves until draw or one player has won.
    - Backpropagation: The result of the simulation is updated back to the root node: simulation/ win/ draw counts are increased accordingly.

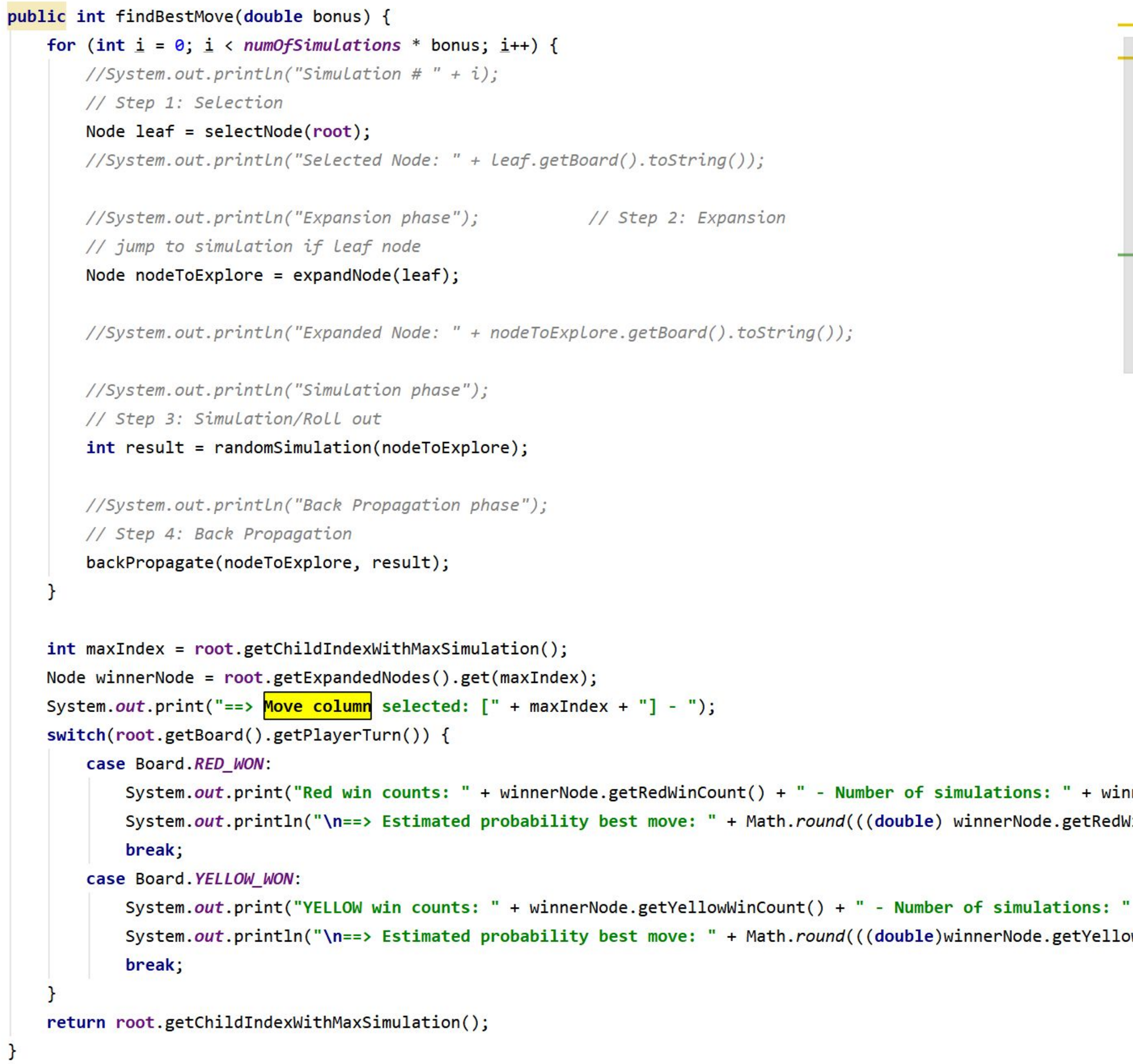


Basic MCTS Algorithm

Figure from Chaslot (2006)

Selection: Recursively pick best node that maximizes UCB for Trees (UCT) as long as the node is visited more than N_0 times
Expansion: Add child node(s) off the selected node to the list of possible nodes we can select in the next round; only 1 node in simplest implementation
Simulation: Randomly simulate game to completion
Backprop: Update nodes on the path with simulation results (wins, number of visits)

- Main function to find best move:



```
public int findBestMove(double bonus) {
    for (int i = 0; i < numOfSimulations * bonus; i++) {
        //System.out.println("Simulation # " + i);
        // Step 1: Selection
        Node leaf = selectNode(root);
        //System.out.println("Selected Node: " + leaf.getBoard().toString());

        //System.out.println("Expansion phase");           // Step 2: Expansion
        // jump to simulation if leaf node
        Node nodeToExplore = expandNode(leaf);

        //System.out.println("Expanded Node: " + nodeToExplore.getBoard().toString());

        //System.out.println("Simulation phase");
        // Step 3: Simulation/Roll out
        int result = randomSimulation(nodeToExplore);

        //System.out.println("Back Propagation phase");
        // Step 4: Back Propagation
        backPropagate(nodeToExplore, result);
    }

    int maxIndex = root.getChildIndexWithMaxSimulation();
    Node winnerNode = root.getExpandedNodes().get(maxIndex);
    System.out.print("==> Move column selected: [" + maxIndex + "] - ");
    switch(root.getBoard().getPlayerTurn()) {
        case Board.RED_WON:
            System.out.print("Red win counts: " + winnerNode.getRedWinCount() + " - Number of simulations: " + winn
            System.out.println("\n==> Estimated probability best move: " + Math.round(((double) winnerNode.getRedWi
            break;
        case Board.YELLOW_WON:
            System.out.print("YELLOW win counts: " + winnerNode.getYellowWinCount() + " - Number of simulations: "
            System.out.println("\n==> Estimated probability best move: " + Math.round(((double)winnerNode.getYello
            break;
    }
    return root.getChildIndexWithMaxSimulation();
}
```

- ***UpperConfidenceTree.java***:
    - Helper class to calculate UCT value for selection phase in MCTS.
        - I modeled it closely to the formula given in the handout in class:

$$\frac{w_i + d_i/2}{n_i} \pm c\sqrt{\frac{\ln N}{n_i}}$$

CS 461 - Program 2 Report
Binh Nguyen - 16168354

Reference:

https://www.youtube.com/watch?v=UXW2yZndl7U
https://towardsdatascience.com/monte-carlo-tree-search-158a917a8baa
https://www.geeksforgeeks.org/ml-monte-carlo-tree-search-mcts/
https://int8.io/monte-carlo-tree-search-beginners-guide/

Note:

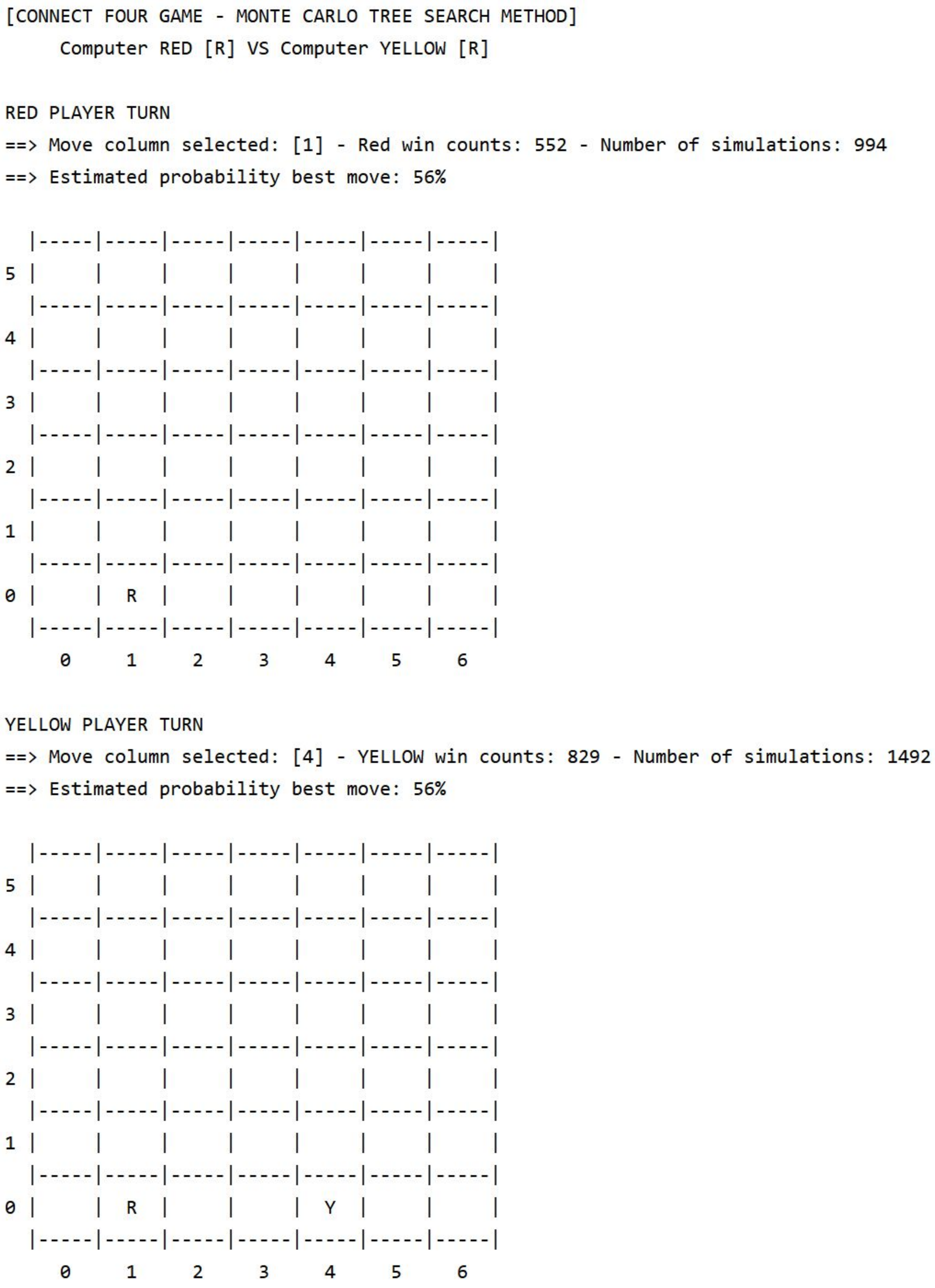
- Increasing the number of simulations & giving Yellow player more number of searches

⇒ Yellow will win more consistently compared to Red.

CS 461 - Program 2 Report
Binh Nguyen - 16168354

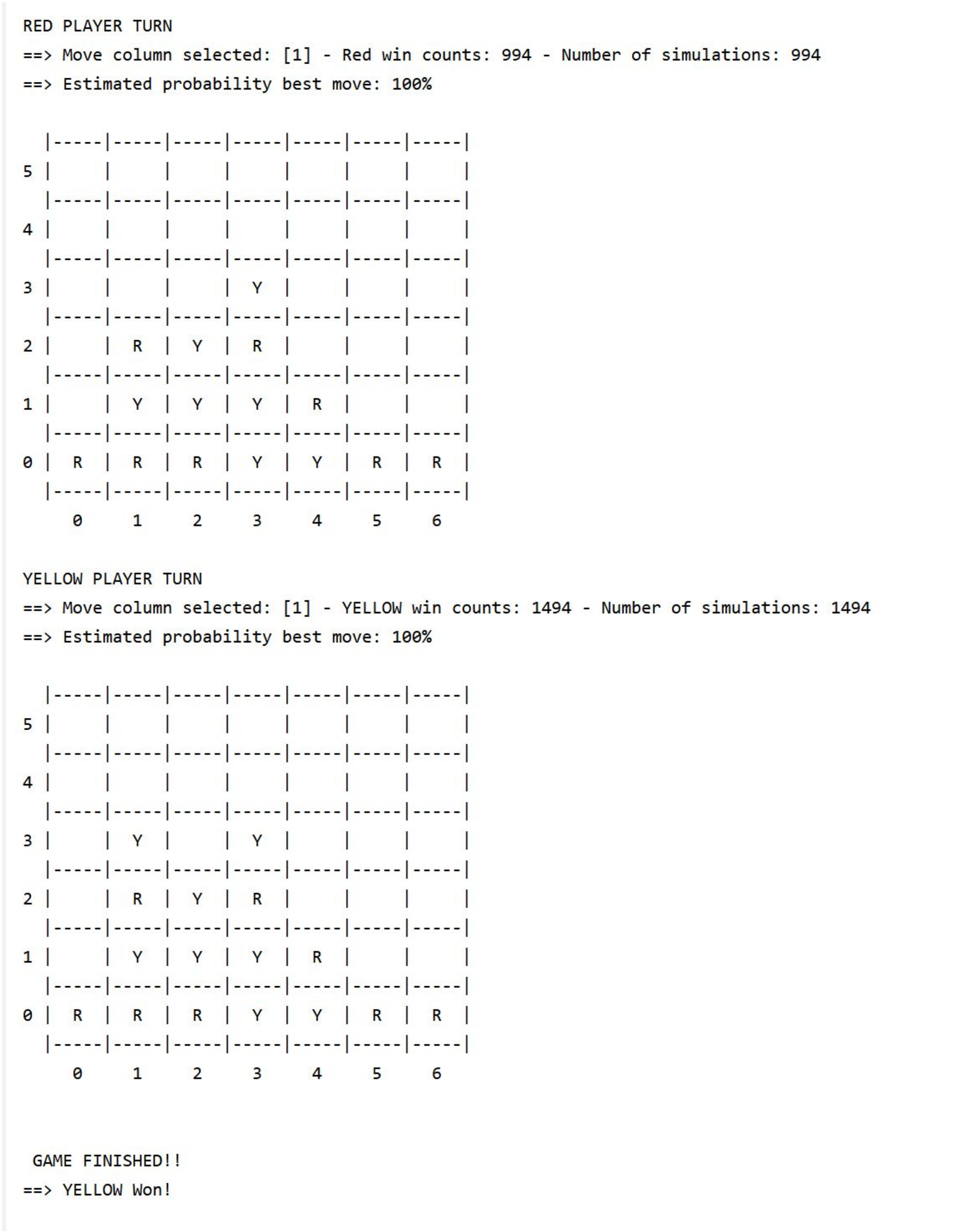
Screenshots:

CS 461 - Program 2 Report
Binh Nguyen - 16168354



```
[CONNECT FOUR GAME - MONTE CARLO TREE SEARCH METHOD]
     Computer RED [R] VS Computer YELLOW [R]

RED PLAYER TURN
==> Move column selected: [1] - Red win counts: 552 - Number of simulations: 994
==> Estimated probability best move: 56%

  |-----|-----|-----|-----|-----|-----|-----|
5 |     |     |     |     |     |     |     |
  |-----|-----|-----|-----|-----|-----|-----|
4 |     |     |     |     |     |     |     |
  |-----|-----|-----|-----|-----|-----|-----|
3 |     |     |     |     |     |     |     |
  |-----|-----|-----|-----|-----|-----|-----|
2 |     |     |     |     |     |     |     |
  |-----|-----|-----|-----|-----|-----|-----|
1 |     |     |     |     |     |     |     |
  |-----|-----|-----|-----|-----|-----|-----|
0 |     |  R  |     |     |     |     |     |
  |-----|-----|-----|-----|-----|-----|-----|
     0     1     2     3     4     5     6

YELLOW PLAYER TURN
==> Move column selected: [4] - YELLOW win counts: 829 - Number of simulations: 1492
==> Estimated probability best move: 56%

  |-----|-----|-----|-----|-----|-----|-----|
5 |     |     |     |     |     |     |     |
  |-----|-----|-----|-----|-----|-----|-----|
4 |     |     |     |     |     |     |     |
  |-----|-----|-----|-----|-----|-----|-----|
3 |     |     |     |     |     |     |     |
  |-----|-----|-----|-----|-----|-----|-----|
2 |     |     |     |     |     |     |     |
  |-----|-----|-----|-----|-----|-----|-----|
1 |     |     |     |     |     |     |     |
  |-----|-----|-----|-----|-----|-----|-----|
0 |     |  R  |     |     |  Y  |     |     |
  |-----|-----|-----|-----|-----|-----|-----|
     0     1     2     3     4     5     6
```

CS 461 - Program 2 Report
Binh Nguyen - 16168354



```
RED PLAYER TURN
==> Move column selected: [1] - Red win counts: 994 - Number of simulations: 994
==> Estimated probability best move: 100%

  |-----|-----|-----|-----|-----|-----|-----|
5 |     |     |     |     |     |     |     |
  |-----|-----|-----|-----|-----|-----|-----|
4 |     |     |     |     |     |     |     |
  |-----|-----|-----|-----|-----|-----|-----|
3 |     |     |     |  Y  |     |     |     |
  |-----|-----|-----|-----|-----|-----|-----|
2 |     |  R  |  Y  |  R  |     |     |     |
  |-----|-----|-----|-----|-----|-----|-----|
1 |     |  Y  |  Y  |  Y  |  R  |     |     |
  |-----|-----|-----|-----|-----|-----|-----|
0 |  R  |  R  |  R  |  Y  |  Y  |  R  |  R  |
  |-----|-----|-----|-----|-----|-----|-----|
     0     1     2     3     4     5     6

YELLOW PLAYER TURN
==> Move column selected: [1] - YELLOW win counts: 1494 - Number of simulations: 1494
==> Estimated probability best move: 100%

  |-----|-----|-----|-----|-----|-----|-----|
5 |     |     |     |     |     |     |     |
  |-----|-----|-----|-----|-----|-----|-----|
4 |     |     |     |     |     |     |     |
  |-----|-----|-----|-----|-----|-----|-----|
3 |     |  Y  |     |  Y  |     |     |     |
  |-----|-----|-----|-----|-----|-----|-----|
2 |     |  R  |  Y  |  R  |     |     |     |
  |-----|-----|-----|-----|-----|-----|-----|
1 |     |  Y  |  Y  |  Y  |  R  |     |     |
  |-----|-----|-----|-----|-----|-----|-----|
0 |  R  |  R  |  R  |  Y  |  Y  |  R  |  R  |
  |-----|-----|-----|-----|-----|-----|-----|
     0     1     2     3     4     5     6


 GAME FINISHED!!
==> YELLOW Won!
```