Checkers Game Report

Mohammad Farrahi
Stud Num: 810198451

Part2:

In this part, we first implemented *getValidMoves* function. In this method, based on what color the piece is, we call *_traverseLeft and _traverseRight* methods with appropriate arguments. If the piece is kingtype, color won't matter and we get all possible moves that can go in four directions.
In *getAllMoves* function, we actually try to get all child states that specific state(in our case game board which makes sense; because board is kind of showing the state of game) can generate which will be used in minimax tree. to reach this goal, for each valid move (coming from *getValidMoves* method) of each piece of given color, we make new board(child board), simulate movement of the piece with *simulateMove* function, and update child board and append it to the list of child boards.

Part3:

evaluate function in this problem, as implemented by default, is difference of number of white and red pieces plus one half of difference of number of white and red kingtype pieces. It seems to be a good evaluation function because we are trying to not lose our pieces and maximize our kingtype pieces. Ii will be positive if whites condition is better, which is

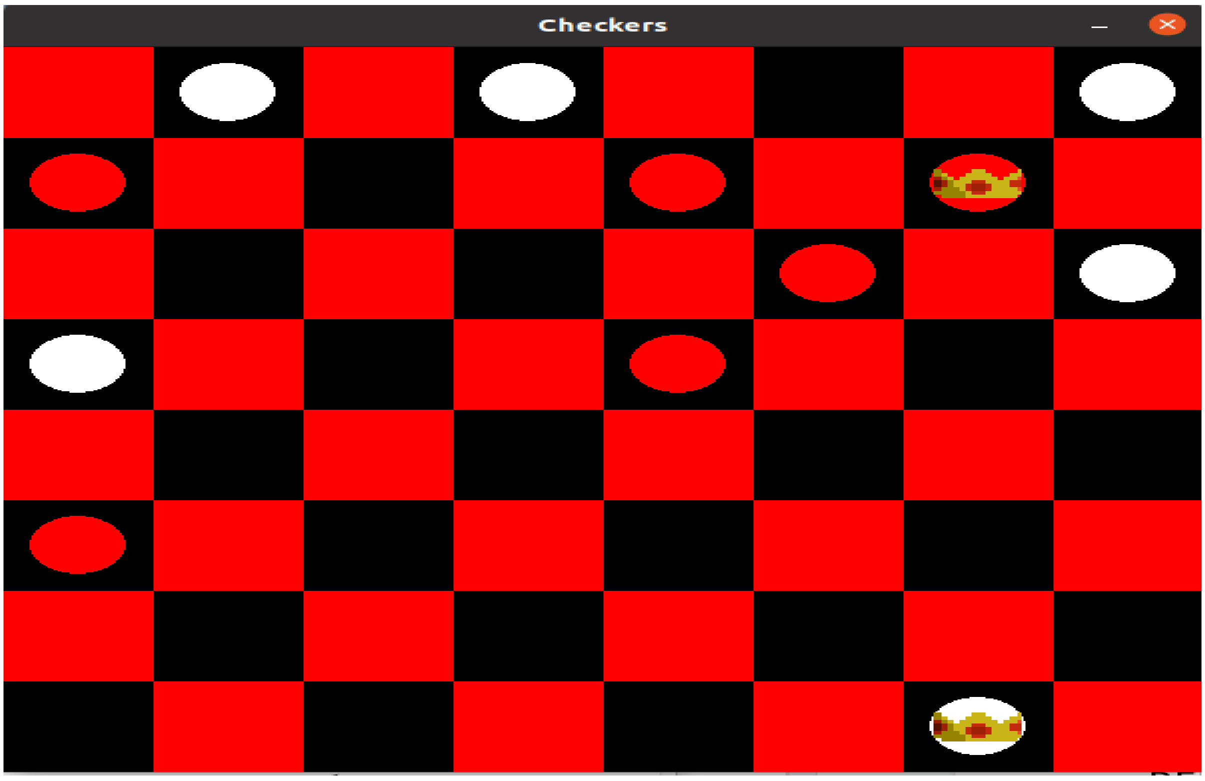
good because in minimax algorithm we consider white a MAX_NODE.
In minimax function we simply implemented a normal recursive minimax algorithm.

Part4:

Game runs very fast and agents don't act very effectively. For example most of red pieces get removed and only one red piece remains. At the end, game goes to a loop state in such a way that there is no effective move in behalf of agents. in this state we 1 kingtype of each color

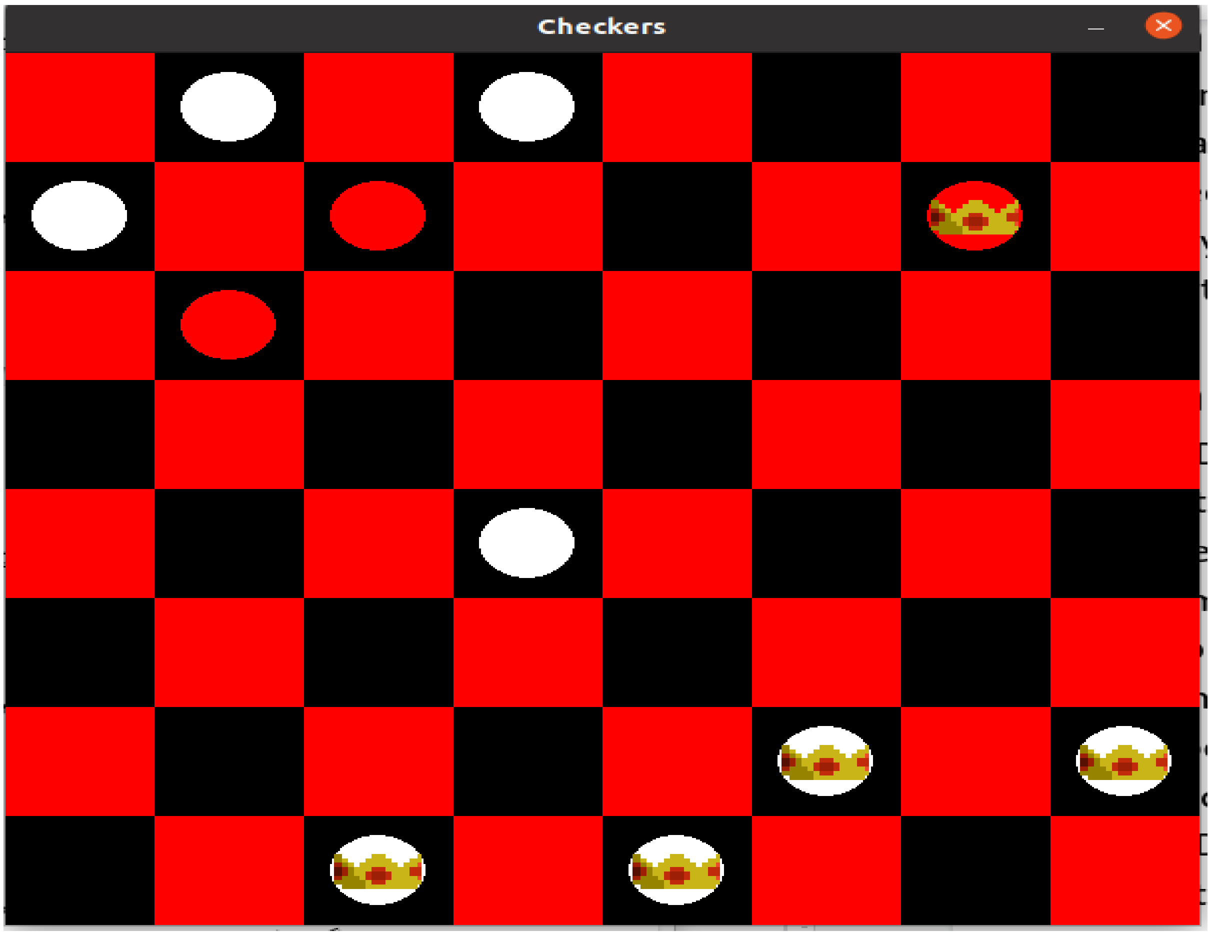
Part5:

RED = 5, WHITE = 2
In this case, minimax computations makes game run slow.; specially red moves; Because it has more depth in its tree. As same as the previous Part, at the end, the game goes to a loop but the number of red pieces are now more than before; Which is a cause of more depth in minimax tree that leads to a better state. Also in this state, we have 1 kingtype of each color.



RED = 2, WHITE = 5

In this case, we have slow movements, specially in white pieces. in this case when game goes to loop state, we have four kingtype of white and one kingtype of red. in overall white has good condition.

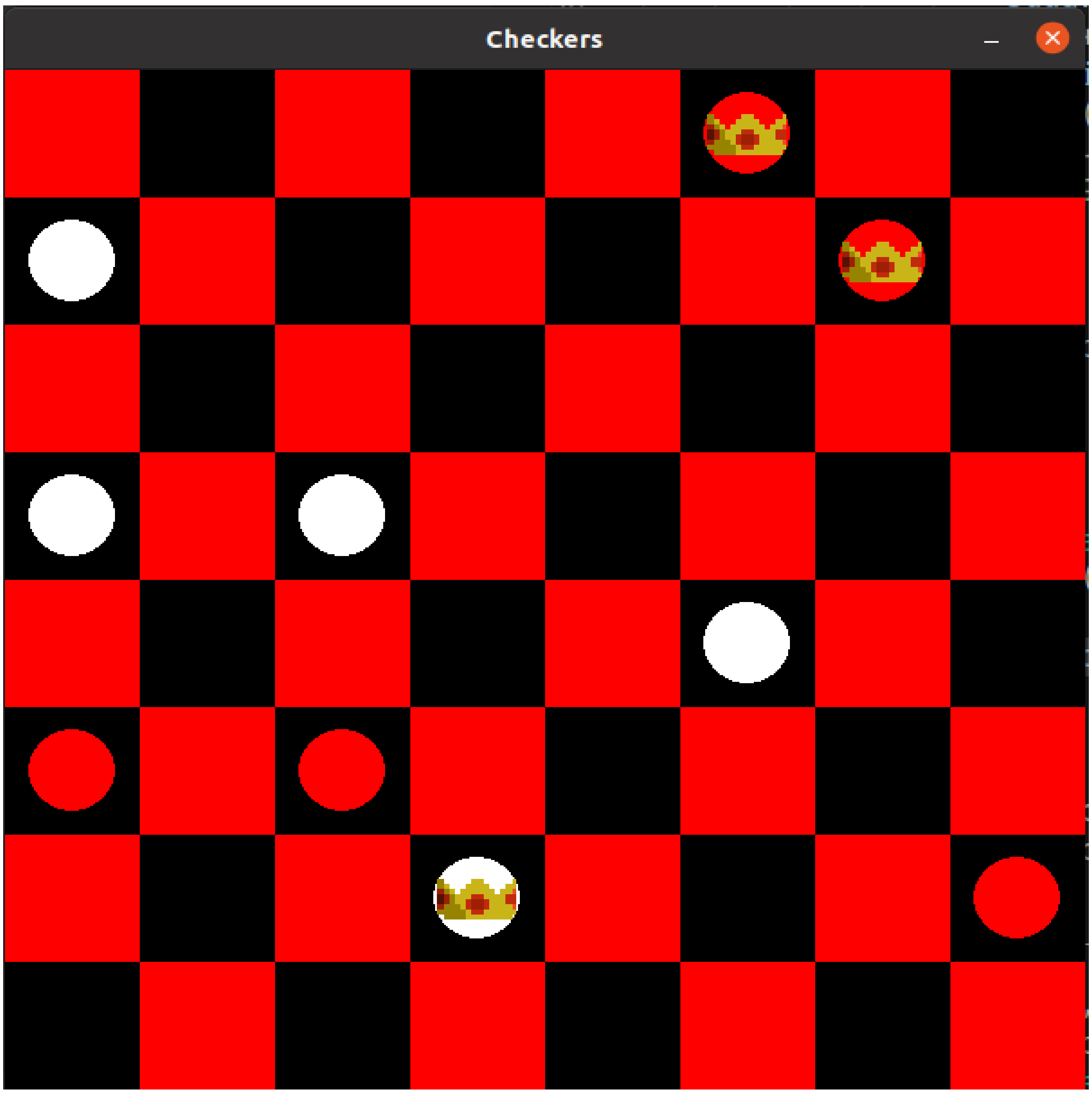


as you can see, white will reach in better result than red is same situation; Perhaps it is matter of who start first in game.

Part6

RED, WHITE = 5

In this case we slow moves from both sides, because depth of both sides are equal and approximately big. As same as previous cases, we finally goes to and in that case, conditions of both sides are nearly equal



*Note: In implementation of *getValidMoves* method, we tried to avoid more than 2 jumps for a piece with giving appropriate arguments to calling methods like *traverseLeft*