GRADER

Name of student running submit: Nathan Lin

Login of student running submit: cs61b-aqt

Second team member's name: Felix Huang

Second team member's login: cs61b-wi

Third team member's name (if any):

Third team member's login:

IMPORTANT: Once you've submitted Project 2 once, the same team member should submit always. If a different teammate must submit, inform cs61b@cory.eecs of all the details. Include a complete list of team members, and let us know which submission you want graded.

If you've submitted your project once, or even written a substantial amount of code together, you may not change partners without the permission of the instructor.

===============================================================================

Does your program compile without errors?

**Yes**

Have you tested your program on the 61B lab machines?

**Yes**

Did you successfully implement game tree search? Did you successfully implement alpha-beta pruning? Are there any limitations on it? What is the default number of search levels set by the one-parameter MachinePlayer constructor (or is it a variable-depth search)?

**We half successfully implemented game tree search. It prunes away some but not all branches. It has limitations in that it still has difficulty leading to a win. The default search depth is 1.**

Describe your board evaluation function in some detail.

**Board evaluation does the following: returns Integer.MAX\_VALUE if the move is a winning move, returns Integer.MIN\_VALUE if the move will stop the opponent from winning, gives a default score of 6 (as opposed to 0) for the first move other either color if the move is around (4,4), adds 3 to the score if you’re on your 6th move and you place the piece in your goal, and gives -2 if you have too many pieces in your goals.**

Does your MachinePlayer use any special method of choosing the first few moves?

**Yes, it picks moves in the center for the first few. This is in our GameTree class.**

Is there anything else the graders should know to help them read your project?

**We used the List package from a previous homework.**

Describe the classes, modules, and interfaces you designed before and while you implemented the project. Your description should include:

**CLASSES DESIGNED**

**GAMEBOARD**

**GAMETREE**

**DList**

**DListNode**

**MODULES**

**DList**

**-Implemented by Nathan Lin from a previous homework**

-It’s a sentinel double sided list (i.e. there’s a prev and a next)

-It extends List

-It’s used in the GameBoard and GameTree class to store the list of valid moves

**DListNode**

**-Implemented by Nathan Lin from a previous homework**

-It’s a node in DList

-It extends ListNode

-It’s used with DList to store data and easily extract data.

**CLASS GAMEBOARD**

**public DList workingMoves(int color)**

**-Implemented by Nathan Lin and Felix Huang**

**-The workingMoves method is used in GameTree class’s minmaxTree method to give the tree a list of valid moves**

-It intakes a color and outputs a DList of legal moves

-It uses two helper functions to create a list of total moves; one function makes a list of add moves and the other makes a list of step moves.

**public boolean network(int color)**

**-Implemented by Felix Huang**

**-The network method is used in GameTree class’s minmaxTree method to determine if a move is a winning move**

-It intakes a color and outputs whether or not that color has a winning network

-It uses helper functions connections, connectionValid, and networkhelper to recursively search for a winning network

**public int evaluate(int color, Move move)**

**-Implemented by Nathan Lin and Felix Huang**

**-The evaluate method is used in GameTree class’s minmaxTree method to determine the best move**

-It intakes a color and a move and outputs a score for the board after the move has been done.

-The scoring process goes as followed: returns Integer.MAX\_VALUE if the move is a winning move, returns Integer.MIN\_VALUE if the move will stop the opponent from winning, gives a default score of 6 (as opposed to 0) for the first move other either color if the move is around (4,4), adds 3 to the score if you’re on your 6th move and you place the piece in your goal, and gives -2 if you have too many pieces in your goals.

**CLASS GAMETREE**

**public GameTree minmaxTree(GameBoard board, DList list\_of\_moves, int color, boolean turn, int alpha, int beta, int searchDepth)**

**-Implmented by Nathan Lin**

**-GameTree utilizes the following methods to output the best move: workingmoves, network, and evaluate**

-It intakes a gameboard and creates a list of valid moves from it. It then runs recursively while intaking the list of moves, color, whose turn it is, an updated alpha, an updated beta, and a search depth that tells the function how many times it should run recursively.

-It returns a good legal move through alpha-beta pruning

-It returns a GameTree object, which holds move data and score data.