Amazons Project

MAT 364 — Fall 2018

Due Date: Friday, November 30

The Amazons Project should follow these guidelines:

Students are required to write basic AI functions for Amazons which implement scope functions for large positions and tree analysis for small positions of height 3 or less. Code should be in C++ and run on the command line. Output will be text-based and so should not include any Windows specific code. C++11 is fine. This text-based game should be playable with human vs AI and should also be able to run as single test case on one position. The AI should implement either Basic Max/Min or Basic Min/Max scope-based AI, and should also have optimal play for small positions of height at most 3.

In order to facilitate use of this code with an existing UI, or to have student AI play against another student AI, the following conventions, as well as further explanations, should be followed:

1. The Amazons Board will be assumed to be 10×10. The board state will be represented by a 10×10 matrix of integers: $0, 1, 2$, or 3, where 0 will represent a blank square, 1 a black (Left) piece, 2 a white (Right) piece, and 3 an X (killed square).
2. Board coordinates are given by row and column, as for a matrix, but with indexing starting at 0 instead of 1. So this means that we start with row 0 on the top and go down to row 9 on the bottom, and we start with column 0 on the left and go up to column 9 on the right. So for example, position $(3, 4)$ is the square in row 3 and column 4.
3. A move consists of seven integers and two matrices:
   - player: (1 for black, or 2 for white)
   - x_0, y_0 coordinates of the square containing the piece to be moved
   - x_1, y_1 coordinates of the destination square to move the piece
   - x_2 ,y_2 coordinates of the square to place the X
   - matrix M_1 for board state before move
   - matrix M_2 for board state after move
4. validateMove(move) is a function that checks to see if a move is valid, returning 1 for a valid move, and 0 for an invalid move.
5. nextMove(board, player) is a function that returns a move on the given board that can be played by the specified player. This function will typically call various AI or random functions in order to generate moves.
6. listMoves(board, player) creates a list of all moves on given board for given player. This is important to implement random player or AI player.

7. StartBoard.txt is a text file with information to give the starting position of a game, typically given as 10 rows with spaces between entries like this:

```
0 0 0 1 0 0 1 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
1 0 0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
2 0 0 0 0 0 0 0 0 2
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 2 0 0 2 0 0 0
```

   A valid move with white to start could be to the resulting board:

```
0 0 0 1 0 0 1 0 0 0
0 0 0 2 3 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
1 0 0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
2 0 0 0 0 0 0 0 0 2
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 2 0 0 0
```

8. TestBoard.txt is a board position which may be used to test an AI on one move. For example, here is a small position, of height 3:

```
3 3 3 3 3 3 3 3 3 3
3 3 3 3 3 3 3 3 3 3
3 3 3 3 3 3 3 3 3 3
3 3 3 3 3 3 3 3 3 3
3 3 3 3 3 3 3 3 3 3
3 3 3 3 3 3 3 3 3 3
3 3 3 3 3 3 3 3 3 3
3 3 3 3 3 3 1 0 3 3
3 3 3 3 3 3 0 3 3 3
3 3 3 3 3 3 0 2 3 3
```

   The value of this position is equal to ± 2 which can be written in Conway's notation as $\{2|-2\}$.

9. A Basic Min/Max scope-based AI does the following: it chooses a move for given board and player which first minimizes opponent's scope, then maximing own scope. To implement this AI, simply form a list consisting of all moves that minimize opponent's scope, then choose from this list a move which maximizes own scope. Note: this AI does not care about other moves, outside the first list, which may achieve a higher value of own scope.

10. A Basic Max/Min scope-based AI does the following: it chooses a move for given board and player which first maximizes own scope, then minimizes opponent's scope. To implement this AI, simply form a list consisting of all moves that maximize own scope, then choose from this list a move which minimizes opponent's scope. Note: this AI does not care about other moves, outside the first list, which may achieve a lower value of opponent's scope.

11. Optimal play for small games of height at most 3 is defined to be game play which achieves a win whenever the player has a winning strategy. For example, the game ± 2 is in outcome class $\mathcal{N}$, so if the AI is playing first (as either Left or Right) on this position, then it should achieve a win. Since such a game has game tree height at most 3, the game tree can be traversed quickly and a win can be found when it exists.
12. Basic text-based game implementation for Amazons, to test and illustrate AI play against human player: this should include command line prompts to get human player moves and respond with AI player moves, printing out the game board after each move. The matrix for the game board can be used, or a slightly more readable version using _ for 0 and X for 3. For example, the starting board and example move from above would be:

```
_ _ _ 1 _ _ 1 _ _ _
_ _ _ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _ _ _
1 _ _ _ _ _ _ _ _ 1
_ _ _ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _ _ _
2 _ _ _ _ _ _ _ _ 2
_ _ _ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _ _ _
_ _ _ 2 _ _ 2 _ _ _
```

and

```
_ _ _ 1 _ _ 1 _ _ _
_ _ _ 2 X _ _ _ _ _
_ _ _ _ _ _ _ _ _ _
1 _ _ _ _ _ _ _ _ 1
_ _ _ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _ _ _
2 _ _ _ _ _ _ _ _ 2
_ _ _ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _ _ _
_ _ _ _ _ _ 2 _ _ _
```

Definition of Scope:

The scope for a given player in a given position is the sum of the scopes of each of that player's pieces. The scope of one piece is calculated as the number of squares which are reachable by that piece as a valid move in the given position (ignoring the part of the move which places an X).

Implementing Scope Functions:

The basic scope function computes the player's scope for each possible move in a given position. This means that the scope is calculated in the new position which would result from the player completing a move, including the placement of the X. If a player has fifty possible moves (many may start with the same piece moving to the same square, but then making a different placement of X) then there are fifty scope calculations to do for fifty new positions. Typically, a move would be chosen which maximize one's own scope, or minimizes the opponent's scope.

To make a better AI one could try to minimize the opponent's scope, then break ties by maximizing one's own scope, or vice versa.

Examples

Find a move for Left which maximizes their own scope from the following position:

•	X		○
			X

We list the new position after each of Left's moves, and their scope in the resulting position:

1. | | | | |
|---|---|---|---|
| X | X | | ○ |
| • | | | X |

Scope: 2

2. | | | | |
|---|---|---|---|
| | X | | ○ |
| • | X | | X |

Scope: 1

3. | | | | |
|---|---|---|---|
| | X | | ○ |
| • | | X | X |

Scope: 2

4. | | | | |
|---|---|---|---|
| X | X | | ○ |
| | • | | X |

Scope: 3

5. | | | | |
|---|---|---|---|
| | X | | ○ |
| X | • | | X |

Scope: 3

6. | | | | |
|---|---|---|---|
| | X | X | ○ |
| | • | | X |

Scope: 3

7. | | | | |
|---|---|---|---|
| | X | | ○ |
| | • | X | X |

Scope: 3

So the highest resulting scope of 3 for Left can be achieved by any of the moves $4-7$. If, additionally, Left wants to minimize the scope of Right, then moves 6 and 7 would achieve this with scope of 1 for Right.

Grading:

For full credit, follow the above guidelines completely. For extra credit, do any of the following:

- combine AI with another student (or students) into a new text-based Amazons game which plays one AI against another.
- increase the optimal play level to height 4 or 5, giving reasons why this works.
- write a function to recognize when the game splits into a sum of two or more games, and implement this to correctly display in text when this happens.
- build a better UI to display output from the text-based game

Here is a useful position of height 3 to illustrate difference between Min/Max, Max/Min, and optimal play:

X		X
○	X	
X		X
•	X	X