Artificial Intelligence Assignment 2 Report

Connect 4 Game

Name: Ahmed Gamal Mahmoud
ID: 18010083

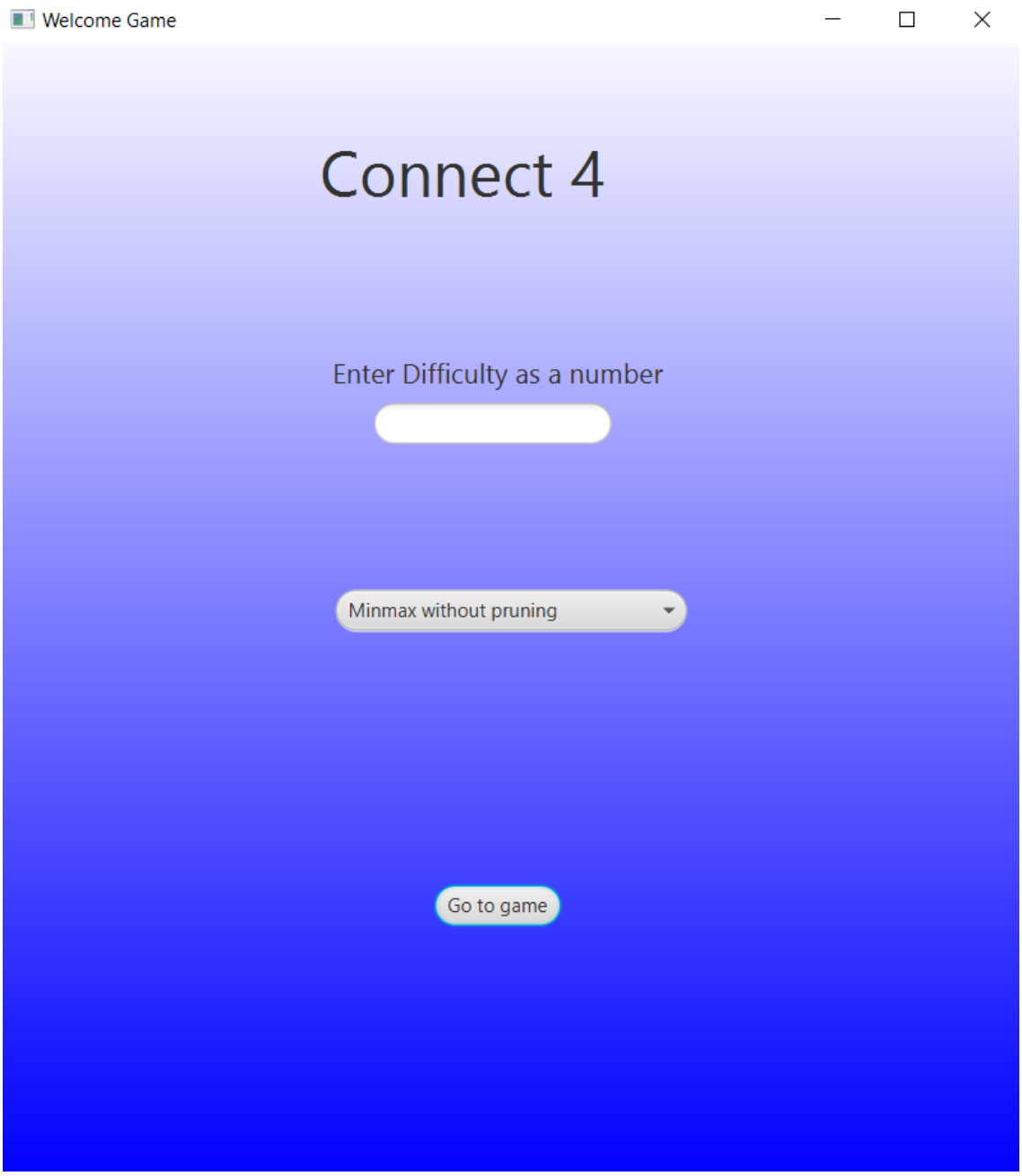
Name: Marwan Mohamed Saad Abougabal
ID: 18011736

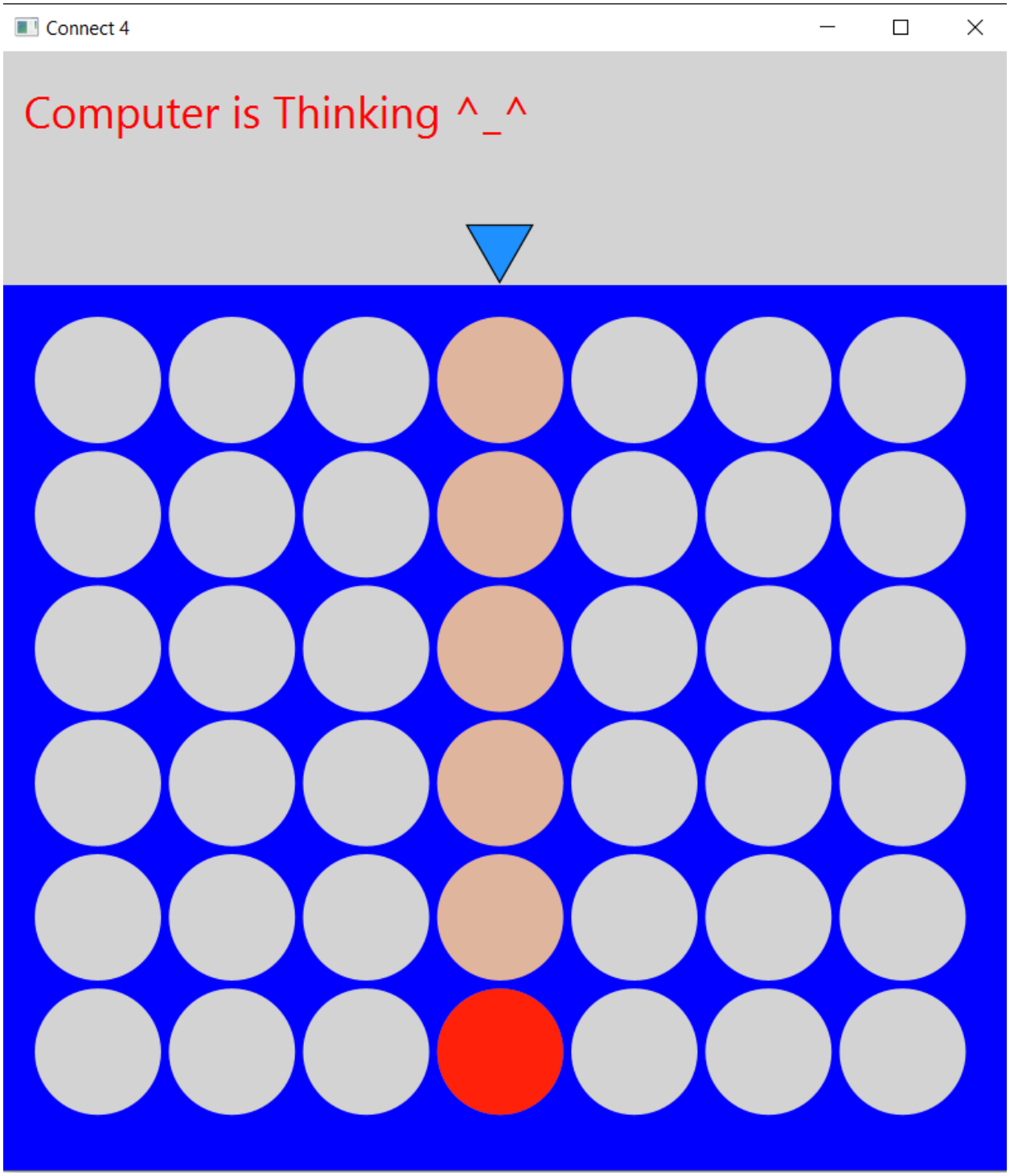
Name: Mohamed Mofreh Abd El-monem El-gazzar
ID: 18011626

Name: Youssef Hany Fathy Shamsia
ID: 18015025

Assumptions and Details

- The human player will start the game with a red disk.
- In the grid the human position takes “1” ,the computer takes “-1” and the empty place takes “0”.
- For this reason Computer will be the Minimizer in the minmax.
- Grid size is always 6 rows x 7 columns.
- Player 2 is the Computer itself.




Connect 4
Computer is Thinking ^_^

GameState

Data Structures

int[]: columnNumber is a one dimensional integer array that holds the count of pieces played in each column.

ArrayList<>: children is an array list of GameState used to hold the children of the current game state.

Algorithms

N/A

GameLogic

Data Structures

N/A

Algorithms

public GameState Initialize_Game(int length, int width,int k, boolean alphaBeta_Pruning, boolean human_isPlayer1): initializes the game.

public ArrayList<GameState> Gen_Children(GameState root): generates children states of current state.

public int[][] buildBoard(GameState state): builds the two dimensional array board from the given game state.

public int calculateHeuristicValue(GameState state): calculates heuristic of given game state.

public int[] calculateScore(GameState state): calculates score of given game state.

Heuristic

Data Structures

int[][]: staticHeuristic is a two dimensional integer array that holds a pre-calculated value for each position in the 6x7 board. Values are used to calculate the primary heuristic.

Algorithms

private int primaryHeuristic(int[][] gameBoard): takes the 6x7 game board as a parameter. It uses the staticHeuristic to calculate the primary heuristic.

private int secondaryHeuristic(int[][] gameBoard): takes the 6x7 game board as a parameter. It checks all chains of pieces on the board and uses the chains to calculate the secondary heuristic based on possible chain extensions (potential moves) and the score of each player on that board. Firstly, the

algorithm checks chains in a row-by-row fashion. For each chain of pieces placed by the same player, the chain length is calculated. The heuristic value is then incremented by the result of the following equation:

value += currentPlayerChain * chainLength * possibleExtensions

where currentPlayerChain is either 1 (for player one / human) or -1 (for player two / computer), chainLength indicates the length of the chain and, possibleExtensions is the number of possible extensions. The number of possible extensions is calculated by checking positions directly adjacent positions in the row and counting them if these moves are legal (there is a piece in the position below or the piece is to be played on the bottom row). If the directly adjacent position is a possible extension, the directly adjacent position to that is also checked in the row.
After checking rows, the columns are checked for chains and the heuristic value is incremented by the same equation shown above. Possible extensions for the column are two positions above the end of the chain. Finally, diagonals are checked starting with positive slope diagonals (bottom-left to top-right) and then negative slope diagonals are checked (bottom-right to top-left). A total of 12 diagonals (6 for each direction) are not checked as they can never form a chain (3 diagonals for each corner of the board). Additionally, for rows, columns, and diagonals, the point scoring chains (chains of

length 4) are multiplied by the score multiplier parameter specified to determine the heuristic of the state based on the scores of the players.

public int calculateTotalHeuristic(int[][] gameBoard): takes the 6x7 game board as a parameter. It calculates the sum of the results of the primaryHeuristic and secondaryHeuristic functions to give the total heuristic value of the current state of the board.

Algorithm

Data Structures

Pair: Used to return the state and it's heuristic value of the terminal state
ArrayList: Used to carry the children of every state
Node : contain the tree nodes of its children of minmax.

Algorithms

GameState decide(GameState state , int k): take the root where you need to calculate it's heuristic at the k level

Pair<GameState, Integer> MAXIMIZE(GameState state,int k,Node root) used to get the maximum heuristic of a child among all the children of the state parameter .. By traversing down the tree until we reach k level and calculate it's heuristic by a pre-defined function and propagate the value

Pair<GameState, Integer> MINIMIZE(GameState state,int k,Node root) used to get the minimum heuristic of a child among all the children of the state parameter .. By traversing down the tree until we reach k level and calculate it's heuristic by a pre-defined function and propagate the value

GameState decidePruning(GameState state , int k): take the root where you need to calculate it's heuristic at the k level but also pruning the not important branches

Pair<GameState, Integer> MAXIMIZE(GameState state,int k,int alpha,int beta,Node root) used to get the maximum heuristic of a child among all the children of the state parameter .. By traversing down the tree until we reach k level and calculate it's heuristic by a pre-defined function and propagate the value but doesn't take into consideration the branches with have alpha >= beta

Pair<GameState, Integer> MINIMIZE(GameState state,int k,int alpha,int beta,Node root) used to get the minimum heuristic of a child among all the children of the state parameter .. By traversing down the tree until we reach k level and calculate it's heuristic by a pre-defined function and propagate the value but doesn't take into consideration the branches with have alpha >= beta

Sample Runs

K = 3

Columns -> 0 , 1 , 2 ,3 , 4 ,5 ,6

Example	Category	MinMax	MinMax(alpha beta)
N/A	Heuristic	N/A	N/A
Player played in column 3 ——————— computer played in column 3	Nodes expanded	400	234
	Running time(μs) With printing tree	121	28
	Running time(μs)	51	10
player played in column 4 ——————— computer played in column 3	Nodes expanded	400	239
	Running time(μs) With printing tree	66	34
	Running time(μs)	25	11
player played in column 3 ——————— computer played in column 3	Nodes expanded	400	220
	Running time(μs) With printing tree	33	51
	Running time(μs)	13	32

K = 5

Columns -> 0 , 1 , 2 ,3 , 4 ,5 ,6

Example	Category	MinMax	MinMax(alpha beta)
N/A	Heuristic	N/A	N/A
Player played in column 3 —————————— computer played in column 2	Nodes expanded	19608	4290
	Running time(μs) With printing tree	728	107
	Running time(μs)	223	39
player played in column 4 —————————— computer played in column 2	Nodes expanded	19608	4595
	Running time(μs) With printing tree	587	71
	Running time(μs)	334	32
player played in column 5 —————————— computer played in column 3	Nodes expanded	19607	6000
	Running time(μs) With printing tree	585	91
	Running time(μs)	325	37

K = 6

Columns -> 0 , 1 , 2 ,3 , 4 ,5 ,6

Example	Category	MinMax	MinMax(alpha beta)
N/A	Heuristic	N/A	N/A
Player played in column 3 —————————— computer played in column 1	Nodes expanded	137256	15415
	Running time(μs) With printing tree	2093	332
	Running time(μs)	772	190
player played in column 4 —————————— computer played in column 5	Nodes expanded	137254	27906
	Running time(μs) With printing tree	1701	367
	Running time(μs)	511	159
player played in column 5 —————————— computer played in column 3	Nodes expanded	137216	12246
	Running time(μs) With printing treee	1968	153
	Running time(μs)	486	93

K = 3.txt , k = 3 alpha.txt ,K = 5.txt , k = 5 alpha.txt ===>
txt files contains tree
of the game.

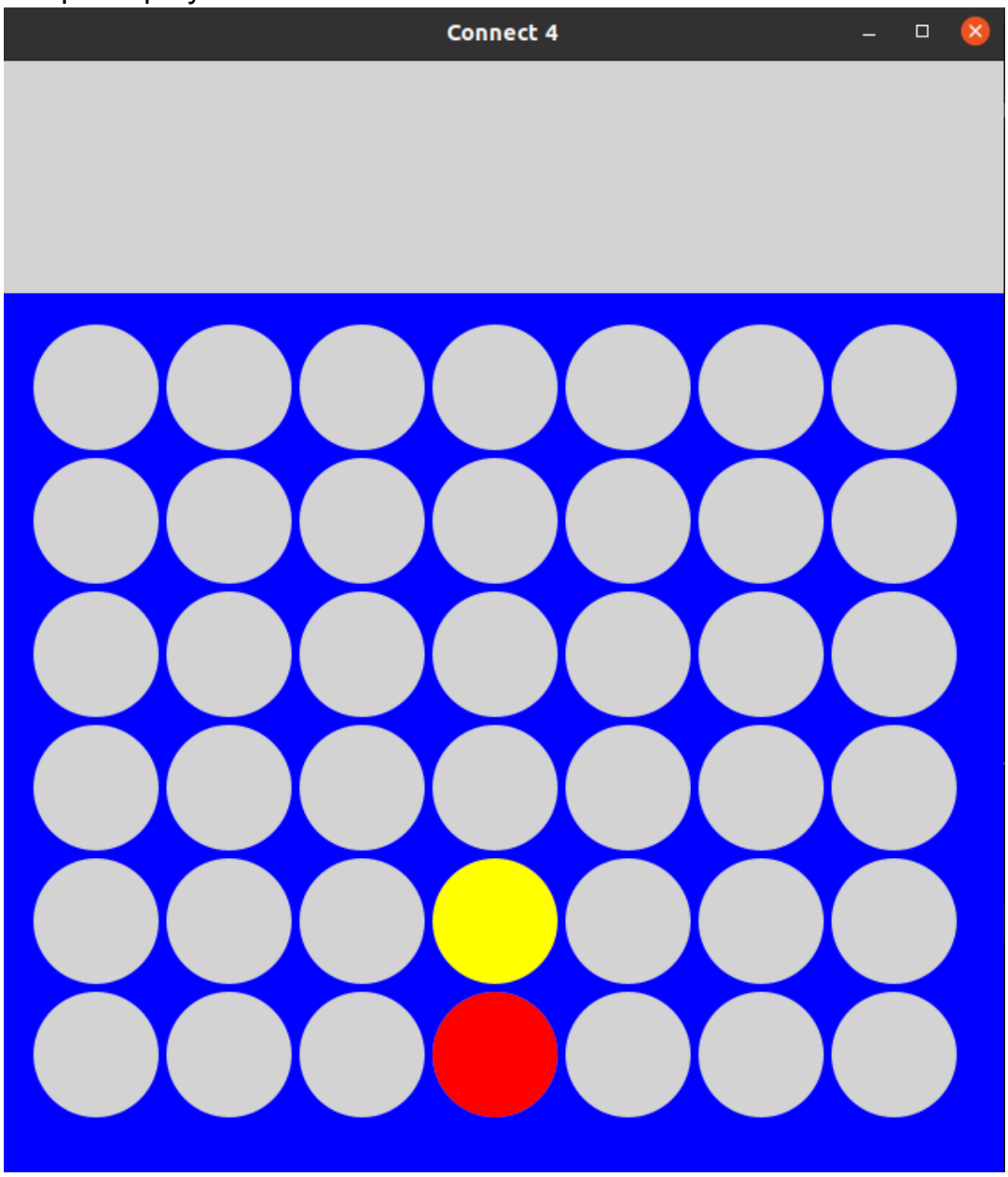
Sample Runs photos

Example : K = 3

Player played in column 3

——————————

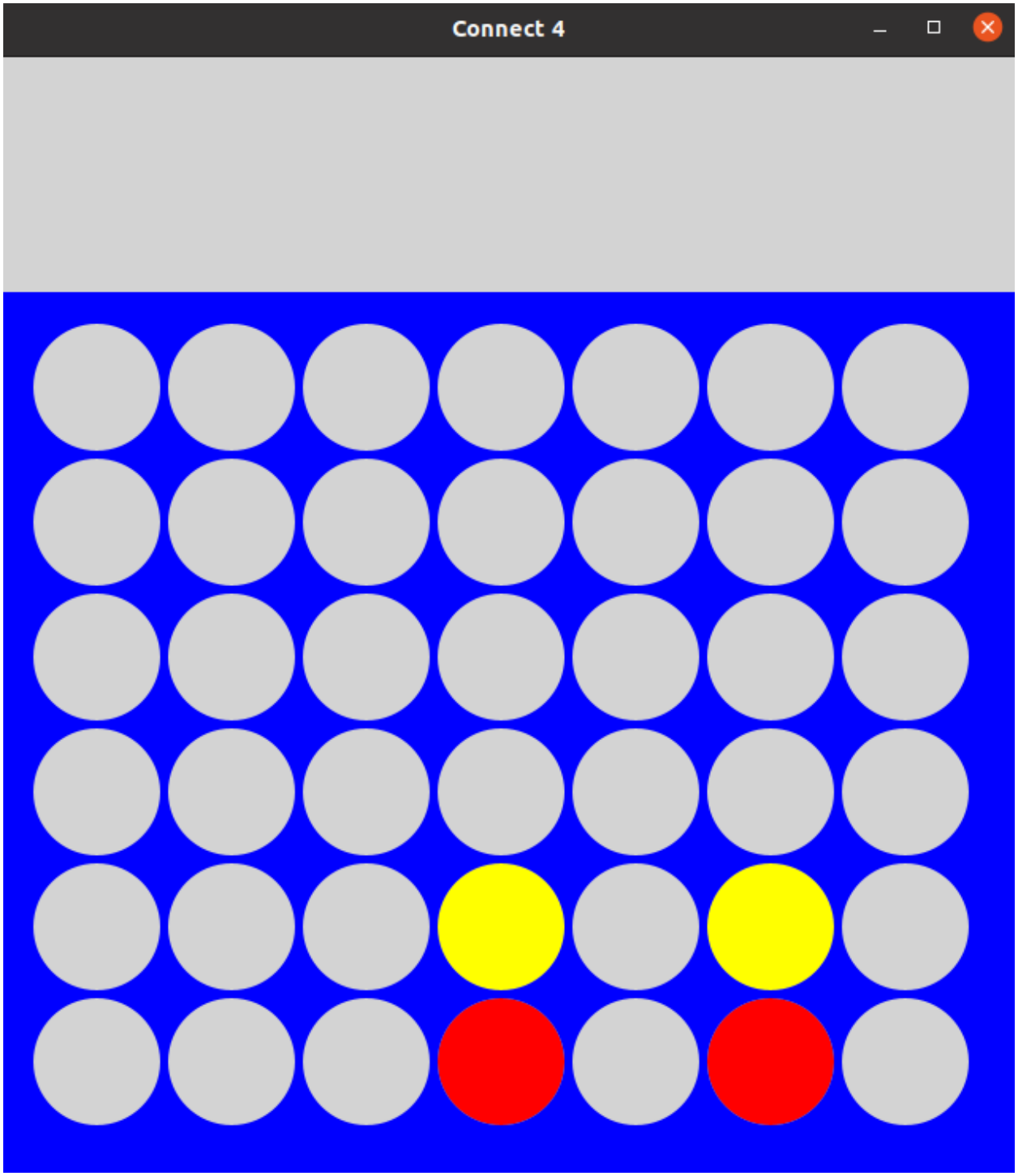
computer played in column 3



Player played in column 5

——————————

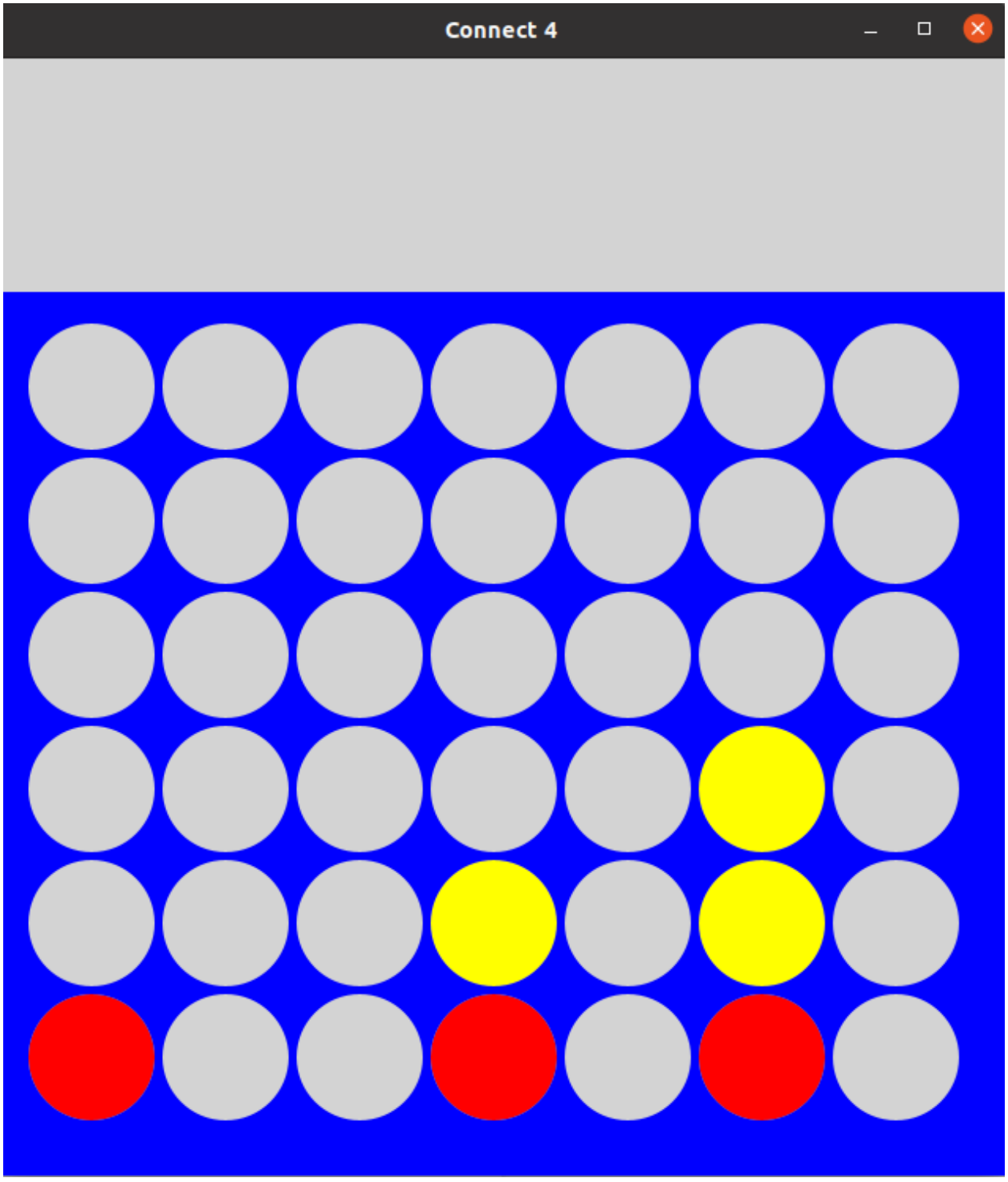
computer played in column 5



Player played in column 0

——----------

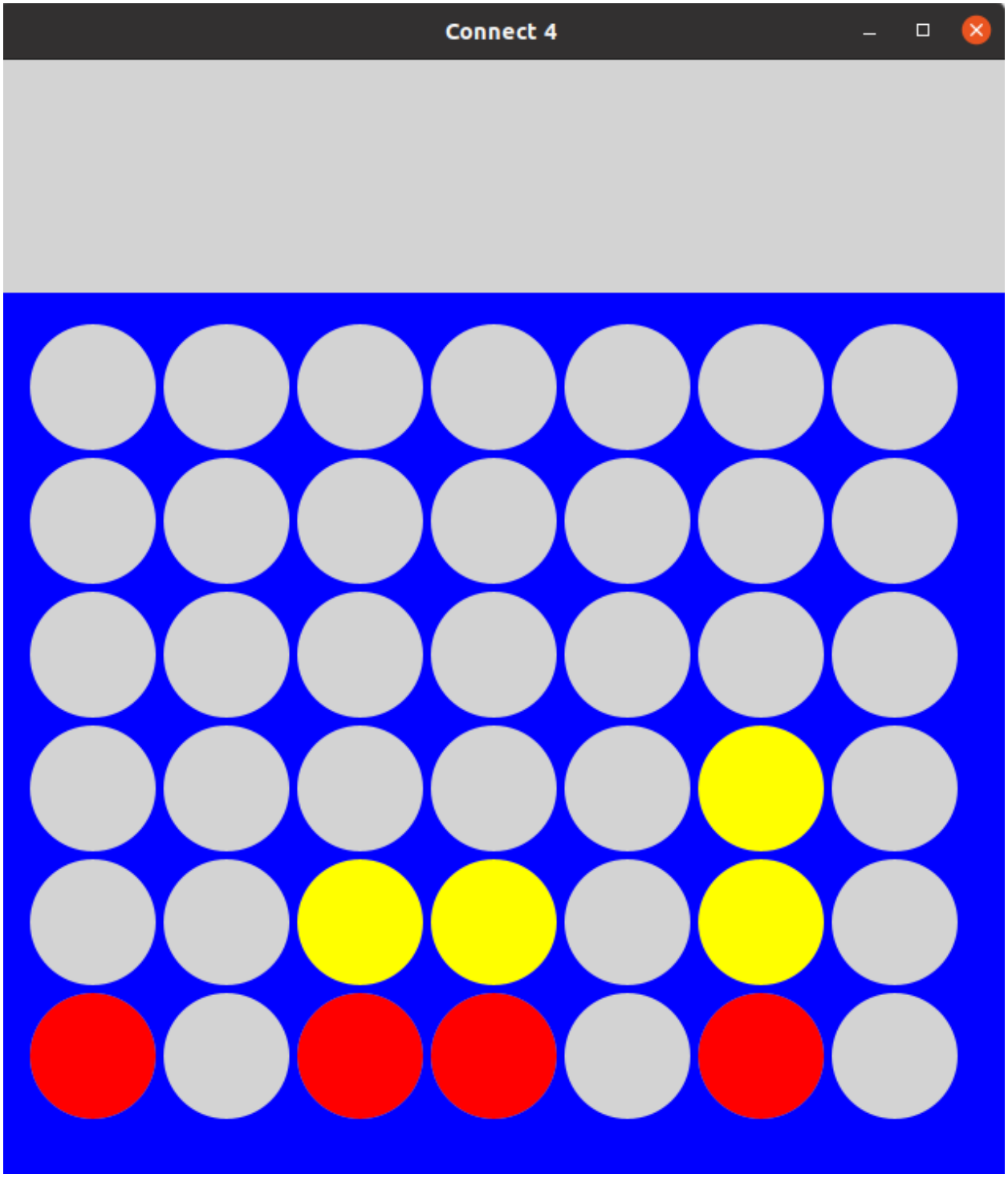
computer played in column 5



Player played in column 2

——————————

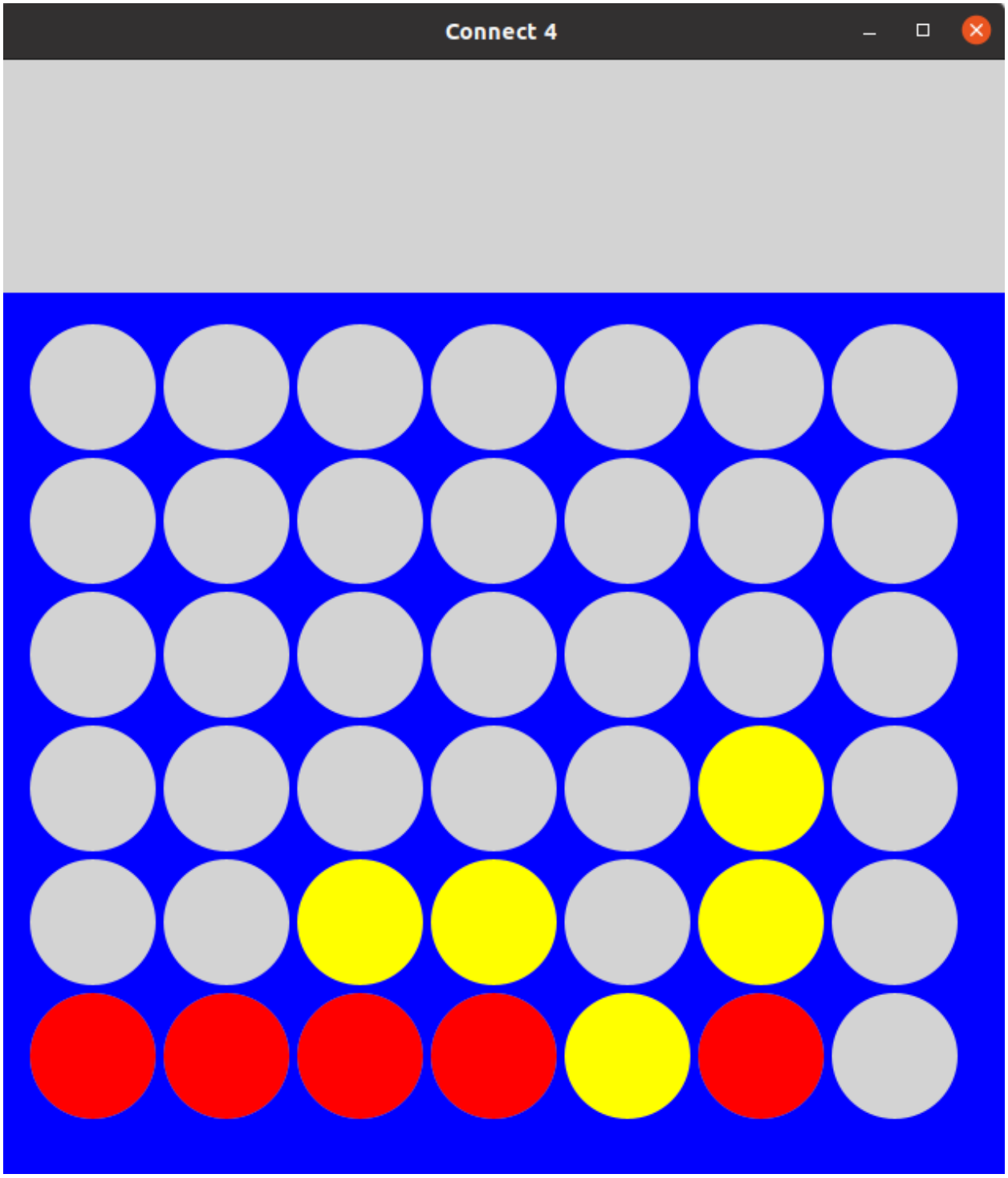
computer played in column 2



Player played in column 1

——————————

computer played in column 4



Game Ended (Computer won)