



CODE WARRIORS 2K21

Brought to you by the Avishkar team

FOR B. TECH 2ND AND 3RD YEAR
AND MCA 1ST AND 2ND YEAR



EVENT COORDINATORS

ARITRA CHATTERJEE

Contact: +916290643054

Teams ID: aritra.chatterjee@mnnit.ac.in

SIDHANT AGARWAL

Contact: +918294167528

Teams ID: sidhant.agarwal@mnnit.ac.in

KSHITIZ SRIVASTAVA

Contact: +919415110060

Teams ID: kshitiz.srivastava@mnnit.ac.in

ANKIT SANGWAN

Contact: +91 7427053874

Teams ID: ankit@mnnit.ac.in

EVENT DESCRIPTION

- Code Warriors is CyberQuest's flagship event under the Artificial Intelligence category. In this event, there will be only one problem statement challenge — write a bot for the game. The game will be a two-player game and will be played by bots against each other. Your bot has to beat the other bot while adhering to the rules of the game.
- Teams will compete each other in multiple knockout rounds which will be held during Avishkar. Each round might have multiple games to ensure fair opportunity to both bots to play as player I. The bot that wins the most rounds will win. In case of a tie, secondary metrics like moves needed to win may be used at the discretion of the coordinators.

Tips for the event:

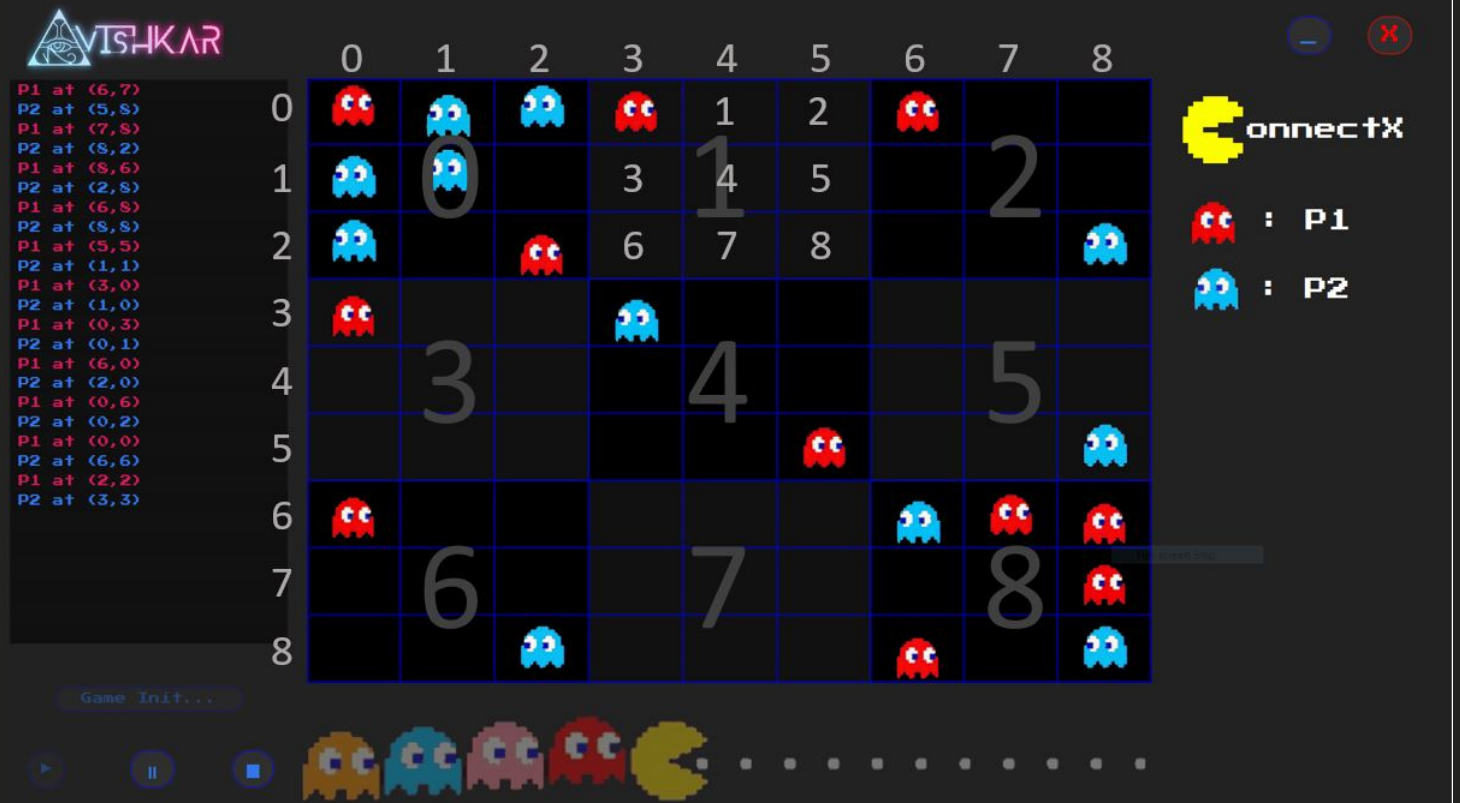
- For beginners, starter agents have been provided which at each point makes a random valid move on the arena
- Additionally, you will also be judged on the basis of code structure and reasoning of the models.
- For any problems, queries, questions, or help at any stage, contact seniors or reach out to peers.

RULES AND REGULATIONS

- This is a team event of maximum 2 members. You can still work individually if you want. Team members can be from any branch. The only prerequisite is that at-least one member of the team should be from the CS/IT branch.
- Cross year teams are not allowed.
- The decision of the judges will be final and binding.
- ANY FORM OF CODE PLAGIARISM WILL NOT BE TOLERATED.



ARENA DESCRIPTION

ULTIMATE CONNECT-X



The game of Ultimate ConnectX is a two-player game inspired from the game of ultimate tic tac toe played on a 9X9 board and hence has similar rules with some minor modifications.

TERMINOLOGIES

- **Main Board:** The main game board consists of a (9 X 9) grid that has been further divided into 9 smaller boards (3 X 3 each) further referred to as local boards.
- **Local Boards:** Each local board for convenience has been assigned a local board number from 0-8 according as shown in the image.
- **Grid square:**
 - **Convention 1:** Each row and column in the grid has been assigned a number from 0-8 as well and hence grid squares are referred to by the tuple (*row_number, column_number*). Further denoted by $(X, Y)_1$
 - **Convention 2:** Each grid space in the local board can also be assigned a number from 0-8 and hence each grid can also be referred by the tuple (*local_board_number, grid_number_in_local*). Further denoted by $(X, Y)_2$
 - The reason for having two conventions is for purposes of convenience and will be evident later. **Note that the point (0,3) in convention 1 translates to (1,0) in convention 2.**
- **History Panel:** The history panel on the right-hand side shows the history of moves performed by both the players.
- **Player Pieces description:**
 - **Player 1:** The grid position occupied by player 1 is denoted by 
 - **Player 2:** The grid position occupied by player 2 is denoted by 

GAME RULES

- The first move is made by player 1. In round 1 and by player 2 in round 2 thus giving both players an equal opportunity to play as player 1.
- The game starts with an active local board of 4. The player 1 can only play in this 3 X 3 local board.
- A move is considered valid if and only if the position of the move made lies within the bounds of the active local board and the position is not occupied already by another piece.
- The active local board for a player is determined by the grid position in local board in which the previous player made the move. eg: If on the first turn, player 1 made a move in the grid $(4,0)_2$ (also denoted as $(3,3)_1$) then since the position played in local board is 0, the active board for the next player would be the 0th local board.
- The only exception to the above is that if local board to be assigned is already filled completely. If this happens, the active board will be the local board with lowest index which has an empty space.
- The objective of the game is to have 4 pieces connected either horizontally, vertically or diagonally before the opponent does.
- The game can end in three states:
 - Win: You managed to connect 4 pieces before the opponent
 - Lose: Your opponent manages to connect 4 pieces before you
 - Draw: All the squares are filled
- The game will be played in form of knockout rounds with other bots in which each round consists of two matches (One as player 1 and another as player 2)
- In both the games are tied or one game win and another loss, other heuristics may be used to determine the winner.
- Last few players remaining may have a round robin instead of knockout and will be at the discretion of the coordinators
- The maximum allowed move time for players is:
 - C++: 1 second
 - Java: 2 seconds
 - Python: 3 seconds

INPUT FORMAT

- The first row consists of how many pieces we have to connect. This value is fixed at 4.
- The next line contains the dimensions of the grid. These values are fixed at 9 9.
- The next 9 lines contain 9 integers each separated by a space which can be either 0, 1 or a 2.
 - 0: Denotes that the grid position is empty.
 - 1: Denotes that the position is occupied by player 1.
 - 2: Denotes that the position is occupied by player 2.
- The next line contains three integers (last_player, x_coordinate and y_coordinate) of move made by last player
 - last_player: Denotes the number assigned to the opponent. Can be used to calculate which pieces are own.
 - x_coordinate: X of (X, Y) (coordinate of the move made by the last player according to convention 1)
 - y_coordinate: Y of (X, Y) (coordinate of the move made by the last player according to convention 1)
- Sample Input:

```
4
9 9
1 2 2 0 0 0 1 0 0
0 0 2 0 0 0 1 0 0
2 2 1 1 0 1 0 0 2
0 0 0 1 0 0 2 0 0
0 0 0 0 0 0 0 0 0
2 2 1 0 0 1 0 0 1
1 0 2 1 0 0 2 2 1
0 0 0 1 0 0 2 2 1
1 2 2 0 0 2 1 1 2
2 8 1
```

OUTPUT FORMAT

- Print two integers: x_coordinate and y_coordinate
 - x_coordinate: X of (X, Y) (coordinate of the move you wish to make according to convention 1)
 - y_coordinate: Y of (X, Y) (coordinate of the move you wish to make according to convention 1)
 - Sample output:
- ```
4 5
```

# INSTRUCTIONS FOR RUNNING THE ARENA

- Before running the build file on the system, ensure that you have Java 8 installed.
- Download the arena from: <https://bit.ly/CodeWarriorsArena>
- Extract the zip file. It contains the jar file for the Arena
- Either open the jar file by double clicking it or running the jar file running the command "*java -jar ConnectX-CodeWarriors.jar*" in terminal
- You can also find the starter agents for the event in the "starter agents" folder
- For doubts, you can contact any of the event coordinators