

# The Students Guide to Cram and the GameLink Platform

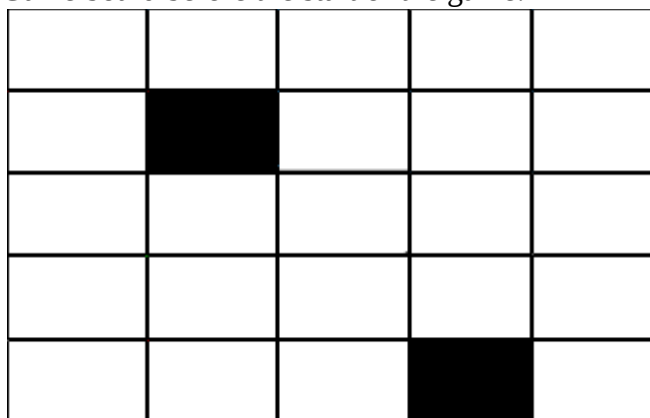
## Rules of Cram:

- Cram is a 2 player turn based game played on a grid (usually 5 by 5).
- Two random cells will be filled at the beginning of the game (shown in black below).
- During their turn the player can place a single 2 x 1 domino on the grid.
- Dominos can be placed either horizontally or vertically, in any empty space on the grid.
- The players take turns placing their dominos with player 1 going first. When a player can no longer make a move; the game is over and the last player to have moved wins.

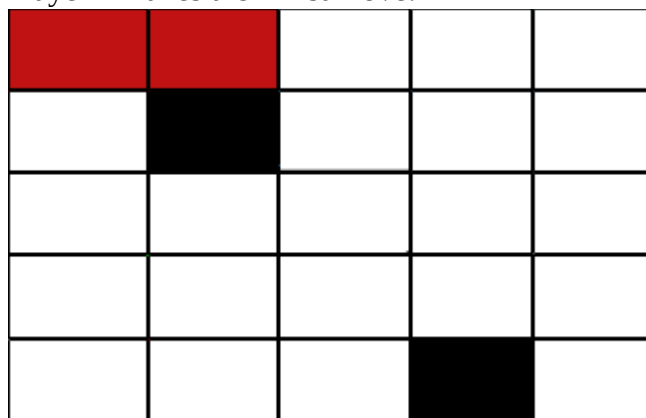
## Score:

The score of the game is determined by counting the number of cells occupied by each player. Additionally, the winner of the match will receive an extra point for every unoccupied cell on the board. Please refer to the Images below for an outline of the different stages of the game played on a 5x5 board

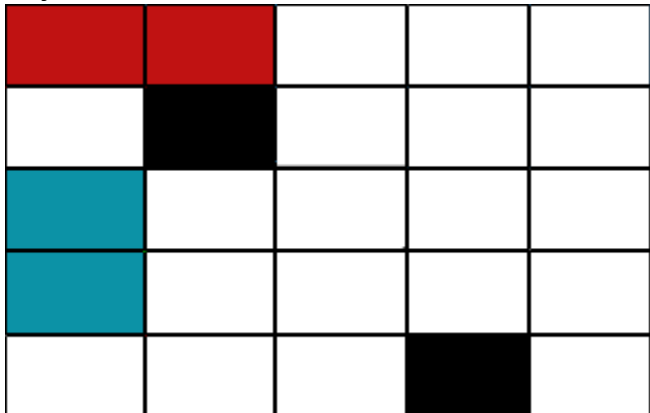
Game board before the start of the game:



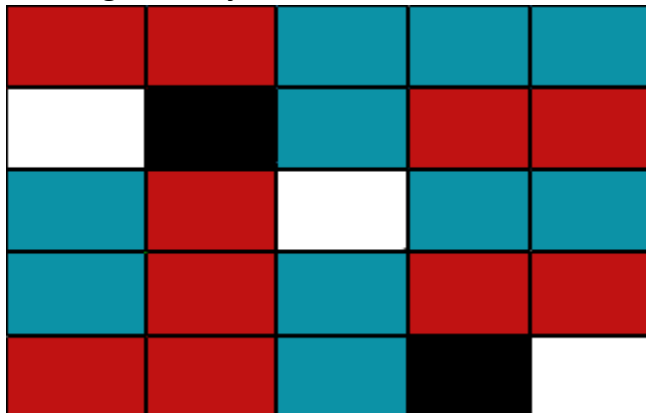
Player 1 makes their first move:



Player 2 makes their first move:



End of game, Player 2 wins

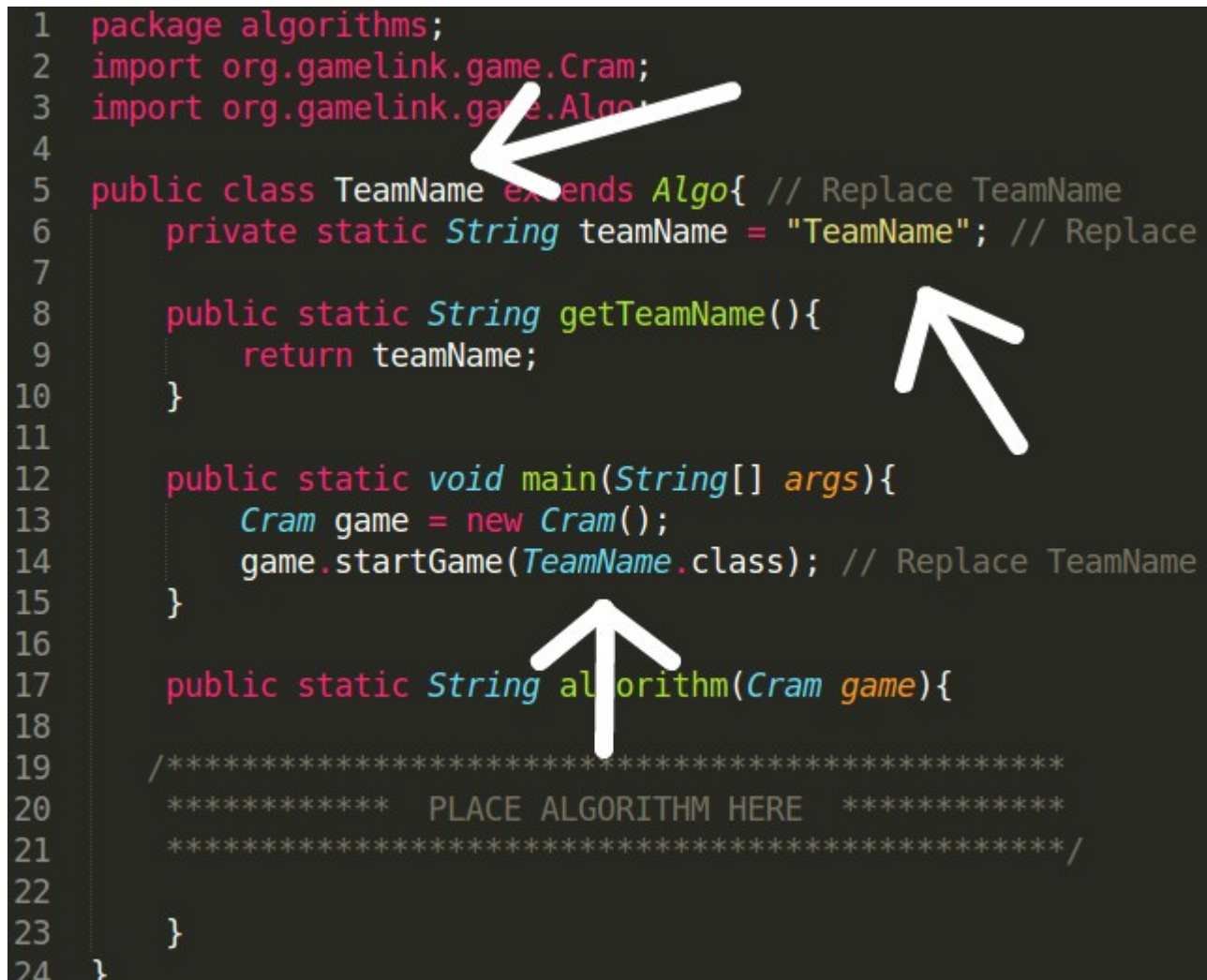


**Final Score:** Player 1: 10, Player 2: 13

## Setting up the GameLink Platform

The gameLink platform is a java program that allows the player's algorithm to play the Cram game with other players on a LAN network. Please follow these steps to set up your algorithm with the GameLink platform

1. Download the Cram zip from <https://github.com/mysteryhobo/Cram> and extract it to the directory of your choosing.
2. Navigate to the algorithms folder and open the TeamName.java file
  - Note: This java source file is where you will be doing all your work for the project
3. Apply the following changes to the java source file
  - On all lines marked "Replace TeamName" (3 total), replace 'TeamName' with your actual team name (Refer to the image below for locations). Your team name must be unique, and must adhere to the restrictions of a java class name (no spaces).
  - Insert your algorithm into the algorithm method (refer to page 4 for more details)
  - Save this new java file with the name of your team followed by .java (Example: Batman.java if your team name was batman)



The image shows a code editor with the following Java code. Three white arrows point to specific locations in the code where 'TeamName' should be replaced with the user's team name:

```
1 package algorithms;
2 import org.gamelink.game.Cram;
3 import org.gamelink.game.Algo;
4
5 public class TeamName extends Algo{ // Replace TeamName
6     private static String teamName = "TeamName"; // Replace
7
8     public static String getTeamName(){
9         return teamName;
10    }
11
12    public static void main(String[] args){
13        Cram game = new Cram();
14        game.startGame(TeamName.class); // Replace TeamName
15    }
16
17    public static String algorithm(Cram game){
18
19        /*****
20         ***** PLACE ALGORITHM HERE *****
21         *****/
22
23    }
24 }
```

Arrows point to the following locations:

- Line 5: `TeamName` in `public class TeamName`
- Line 6: `TeamName` in `private static String teamName = "TeamName";`
- Line 14: `TeamName` in `game.startGame(TeamName.class);`

## Compiling and Running the Game:

The following steps show how to compile and run the Cram game using the command line, if you wish to use an IDE, please refer to your IDE's manual for information on importing a project.

### Compiling and Running Using Command Line:

1. Open the command Line (cmd for windows, terminal for linux).
2. Using the change directory command (cd <filename>) change the command line directory to the Cram-master folder that you downloaded.
3. Once in the Cram-master directory you can enter the following command to compile your algorithm. Remember to replace TeamName.java with your actual algorithm java file name that you made in step 3 of the previous page.

```
javac -cp Cram.jar algorithms/TeamName.java
```

4. You may now run your algorithm with the game by entering the following command. Remember to replace TeamName with your actual algorithm java file name

```
Windows: java -cp Cram.jar;. algorithms.TeamName
```

```
Linux: java -cp Cram.jar:. algorithms.TeamName
```

If successful you should see something like the image below:

```
*****
CONFIG: Inital Cells Filled = 2
CONFIG: Board Height = 5
CONFIG: Board Width = 5
CONFIG: Time Limit = 30
CONFIG: Time Delay = 3
CONFIG: Display Board = true
*****
Please enter your player number <1 or 2>:
```

## Connecting:

The Gamelink platform connects players via a peer to peer connection over a local area network(LAN). The following instructions outline how to connect as each player:

### Connecting as Player1:

Run the game, enter "1" to identify as player 1, and wait for player2 to connect (Player1's IP address is then displayed in the command prompt).

### Connecting as Player2:

Run the game and enters "2" to identify as player 2 and enter the IP address of Player1 (example: 10.20.64.76)

## Coding your algorithm:

Your algorithm will return a String containing 4 numbers, the first two numbers represent the y and x coordinates of the first cell of the domino you wish to place and the second two numbers represent the y and x coordinates of the second cell of the domino. Example: the String 4041 corresponds to a horizontally placed domino in the bottom left corner of the board. See the image above for the coordinates of every cell on a 5 x 5 board.

Y = 0, X = 0	Y = 0, X = 1	Y = 0, X = 2	Y = 0, X = 3	Y = 0, X = 4
Y = 1, X = 0	Y = 1, X = 1	Y = 1, X = 2	Y = 1, X = 3	Y = 1, X = 4
Y = 2, X = 0	Y = 2, X = 1	Y = 2, X = 2	Y = 2, X = 3	Y = 2, X = 4
Y = 3, X = 0	Y = 3, X = 1	Y = 3, X = 2	Y = 3, X = 3	Y = 3, X = 4
Y = 4, X = 0	Y = 4, X = 1	Y = 4, X = 2	Y = 4, X = 3	Y = 4, X = 4

### Algorithm Rules:

- Any String returned by the algorithm not following the standard outlined above will result in an invalid move and an immediate loss of the match.
- Attempting to place a domino on an already occupied cell or a cell outside the bounds of the game board will result in an invalid move and an immediate loss of the match.
- Any algorithm that exceeds the time specified by the move time limit setting will result in an invalid move and an immediate loss of the match

### Game.getBoard();

Your algorithm can call the game.getBoard() method to return the current configuration of the gameBoard in the form of a 2d integer array. Where 1 represents moves made by player 1, 2 represents moves made by player 2, and 6 represents the randomly added cells at the beginning of the game. The table to the right out is an example of a 2d integer array that the getBoard method returns. Below lies an example of how the method should be used

1	1	2	2	2
0	6	2	1	1
2	1	0	2	2
2	1	2	1	1
1	1	2	6	0

```
int[][] gameBoard = game.getBoard()
```

**Note:** Do not alter any of the pre-written code as it may cause errors when marking your algorithm. You can however, add your own imports and methods to be called from within you algorithm method.

## Game Settings:

While testing your algorithm you may want to change some of the game settings. This can be done by changing the values in the Cram.properties file. The following is a list of all the properties.

- **InitialCells:** The InitialCells property dictates the number of cells that will be randomly filled in before the start of the game (Normally 2).
- **BoardHeight:** The BoardHeight property dictates the height of the board (normally 5).
- **BoardWidth:** The BoardWidth property dictates the width of the board (normally 5).
- **MoveTimeLimit:** The MoveTimeLimit property dictates the maximum amount of time (in seconds) your algorithm can take to move. Entering a value of 0, will turn off the time limit
- **DisplayBoard:** The displayBoard property dictates whether or not the board will be displayed after each move.
- **MoveDelay:** The MoveDelay property adds a delay (in seconds) between moves therefore slowing the game, allowing users to follow the playing of the game

**Note:** Ensure these properties are the same as your opponent's as discrepancies will cause errors during the playing of the game

## Playing Against yourself:

If you wish to play your algorithm against itself, or play different algorithms against each other for comparison purposes, you can do so using the following steps:

### Against itself:

To play the algorithm against itself simple run the algorithm in two different command lines, one as player 1, the other as player 2 and follow the instructions outlined in the Connecting section

### Creating another algorithm to play against:

1. Create a copy of your current algorithm java source file.
2. Repeat the changes conducted in step 3 from Setting up the Gamelink platform section but with a different team name (team name must be different than the one used previously, but still must adhere to the same java class naming restrictions).
3. Compile and run both algorithms in separate command lines, one as player1 the other as player2

## The Tournament and submitting your algorithm:

A round-robin style tournament will be conducted to determine the strength of the algorithms. Each algorithm will face off against all others twice; once as player 1 and once as player 2 to ensure fairness. Algorithms will be ranked in order of points earned, where winning a match awards 2 points. In the event of a tie in points, the sum of all game scores will act as the tie breaker.

**Submission:** To submit your algorithm simply upload your .java file to blackboard, be sure to only upload the .java file and not the whole Cram-master folder.

**NOTE:** Do not alter any of the pre-written code. If you submit a .java file with altered pre-written code (other than that specified in the Setting up the GameLink platform section) your algorithm will not be runnable in the tournament, resulting in a loss of every match and a score of zero.

If you have any questions you can email me at: [peter.little1@uoit.net](mailto:peter.little1@uoit.net)