Tic-Tac-Toe

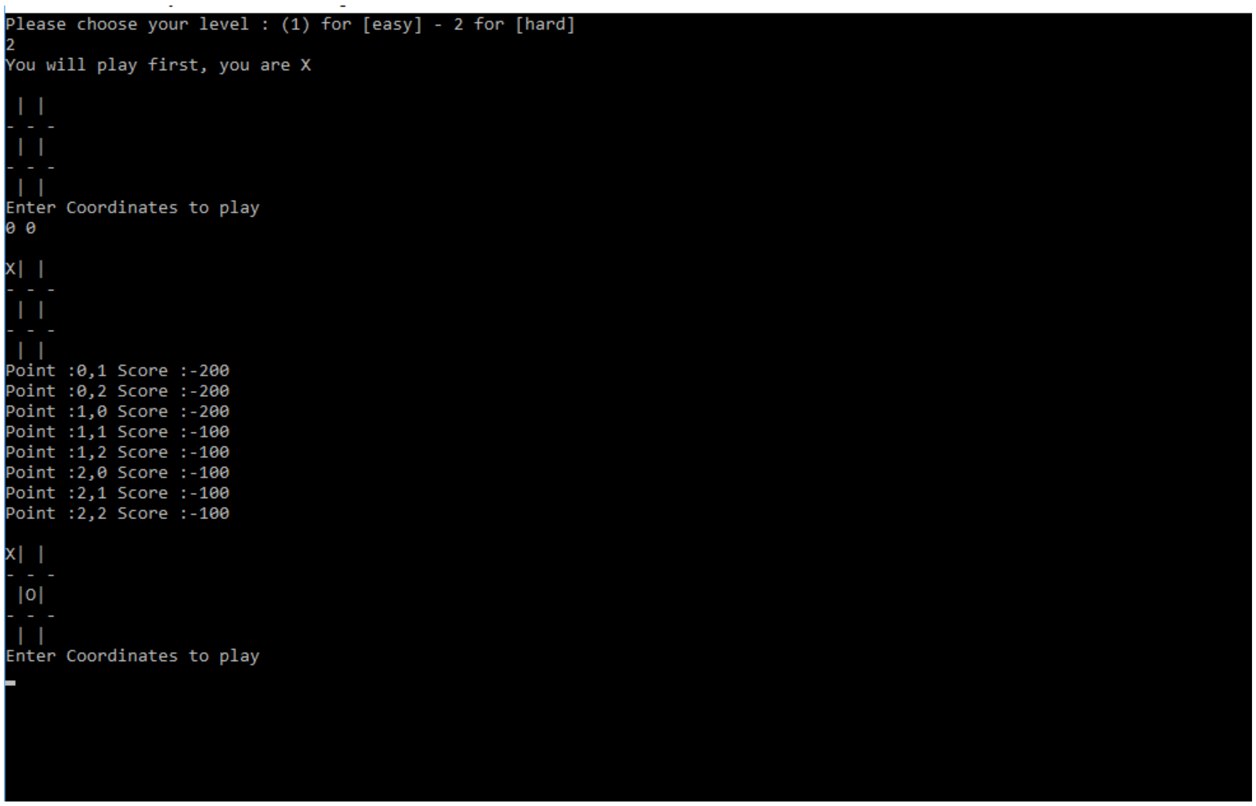
Algorithm used in steps :

1- Will have to variables at the beginning:
   Alpha; which is the max lower bound of solutions (maximizer can update this value)
   Beta; which is the min upper bound of possible solutions (minimizer can update this value),
   Alpha initially will be set to – infinity and Beta initially will be set to + infinity .
2- When it's the computer turn to choose a slot to play in we will call a function names alpha_beta(), which takes some arguments and return sets the optimal move of the computer to play in, this function takes the initial alpha and beta we have, turn : who's gone play the human or the computer and the depth of the traversed tree which is initially set to 0 as we always start from a root node.
3- alpha_beta() function: we will begin traversing all possible solutions we have to choose the optimal move but we will prune some leaves as we traverse as we no longer need to see the value of these leaves and this what alpha and beta gives us.
4- And so we begin traversing the whole solutions tree depth first (DFS) until a child node has been reached.
5- After each child node we update parent alpha and beta.
6- We traverse other siblings only if beta<alpha other than that we prune that sibling
7- If alpha>=beta we must return a value to this sibling node either +infinity or – infinity in case the parent is a minimizer or a maximizer respectively.
8- Other than that we return a heuristic number to that node in that path.
9- The last step when we return from (DFS) to the root node we started from we would have heuristic data for all paths and so we will return the best solution we got from all paths, in our case the computer (O) is a maximizer so the best solution we got is the max of all heuristic data we have for all the children paths and this will guarantee to take the best path to win in fewest number of moves so if the computer have a shot to win he will take it.

Cases :

For the hard difficulty :

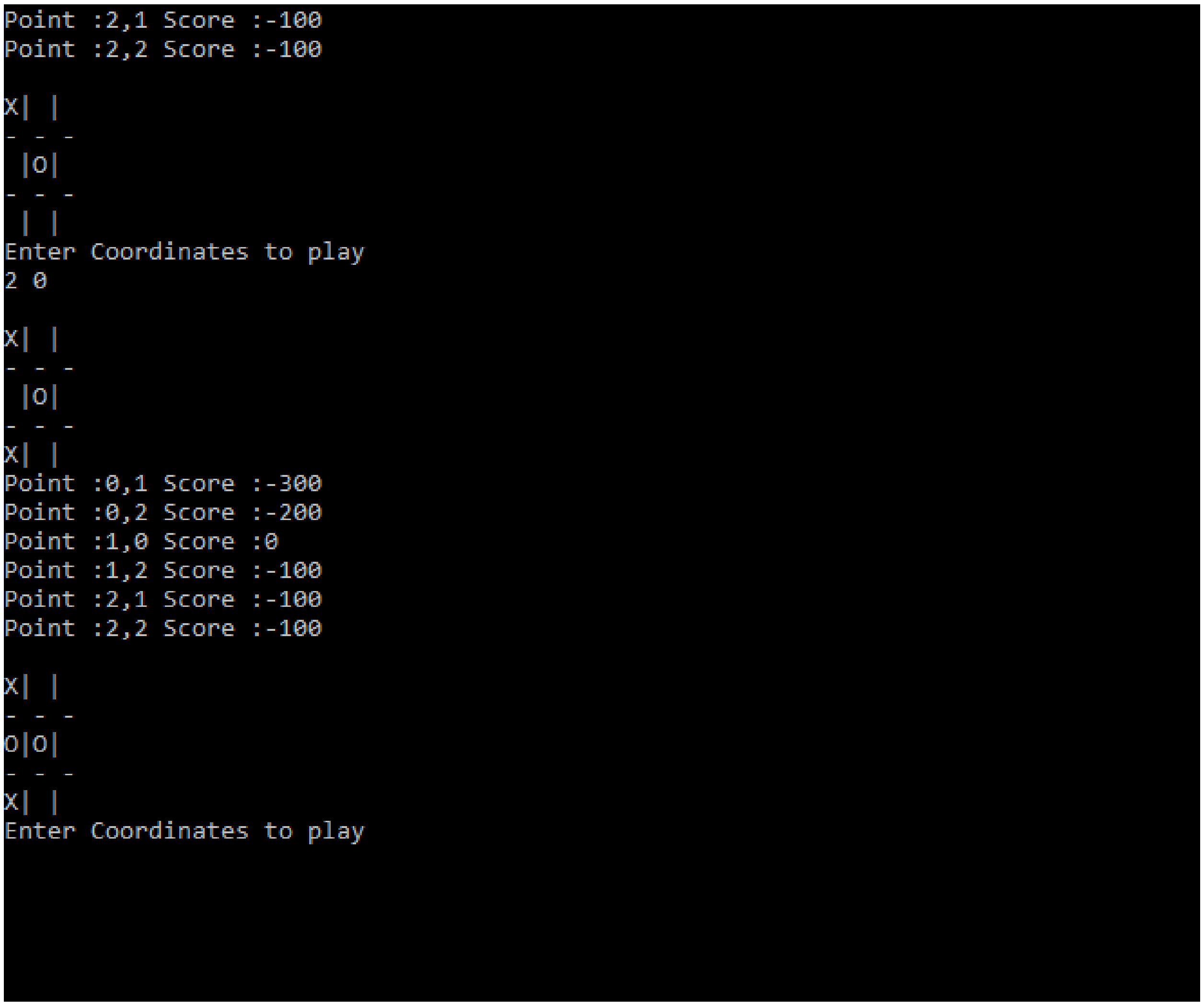
- Human plays @ [0-0]



```
Please choose your level : (1) for [easy] - 2 for [hard]
2
You will play first, you are X

 | |
- - -
 | |
- - -
 | |
Enter Coordinates to play
0 0

X| |
- - -
 | |
- - -
 | |
Point :0,1 Score :-200
Point :0,2 Score :-200
Point :1,0 Score :-200
Point :1,1 Score :-100
Point :1,2 Score :-100
Point :2,0 Score :-100
Point :2,1 Score :-100
Point :2,2 Score :-100

X| |
- - -
 |O|
- - -
 | |
Enter Coordinates to play
_
```

We clearly see here than the computer has 8 paths to choose from each will return a score to him so he will choose the best value which is the maximum score as the computer is always the maximizer, and so he plays @ [1,1]

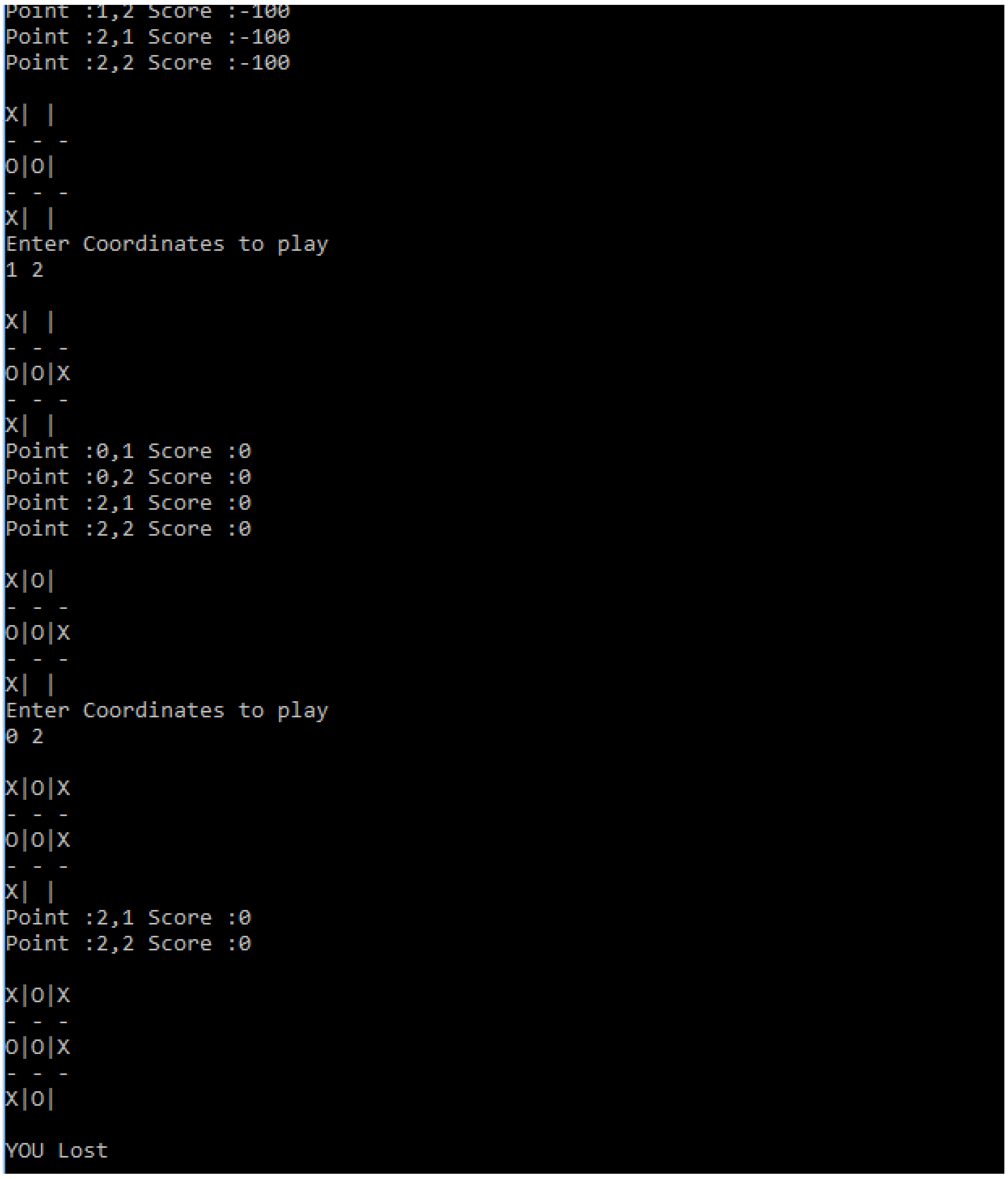
- Human plays @ [2-0]



```
Point :2,1 Score :-100
Point :2,2 Score :-100

X| |
- - -
 |O|
- - -
 | |
Enter Coordinates to play
2 0

X| |
- - -
 |O|
- - -
X| |
Point :0,1 Score :-300
Point :0,2 Score :-200
Point :1,0 Score :0
Point :1,2 Score :-100
Point :2,1 Score :-100
Point :2,2 Score :-100

X| |
- - -
O|O|
- - -
X| |
Enter Coordinates to play
```

And here the computer have 6 paths each will return a specific score so he choose the one with the maximum value which is @[1,0], and so he blocked the opponent move to win .

- Human Plays @[1-2] to block opponent



```
Point :1,2 Score :-100
Point :2,1 Score :-100
Point :2,2 Score :-100

X| |
- - -
O|O|
- - -
X| |
Enter Coordinates to play
1 2

X| |
- - -
O|O|X
- - -
X| |
Point :0,1 Score :0
Point :0,2 Score :0
Point :2,1 Score :0
Point :2,2 Score :0

X|O|
- - -
O|O|X
- - -
X| |
Enter Coordinates to play
0 2

X|O|X
- - -
O|O|X
- - -
X| |
Point :2,1 Score :0
Point :2,2 Score :0

X|O|X
- - -
O|O|X
- - -
X|O|

YOU Lost
```

So the computer tries to get the advantage of heuristics to go for a win and he plays @ [1,1] which guarantee to him the max value

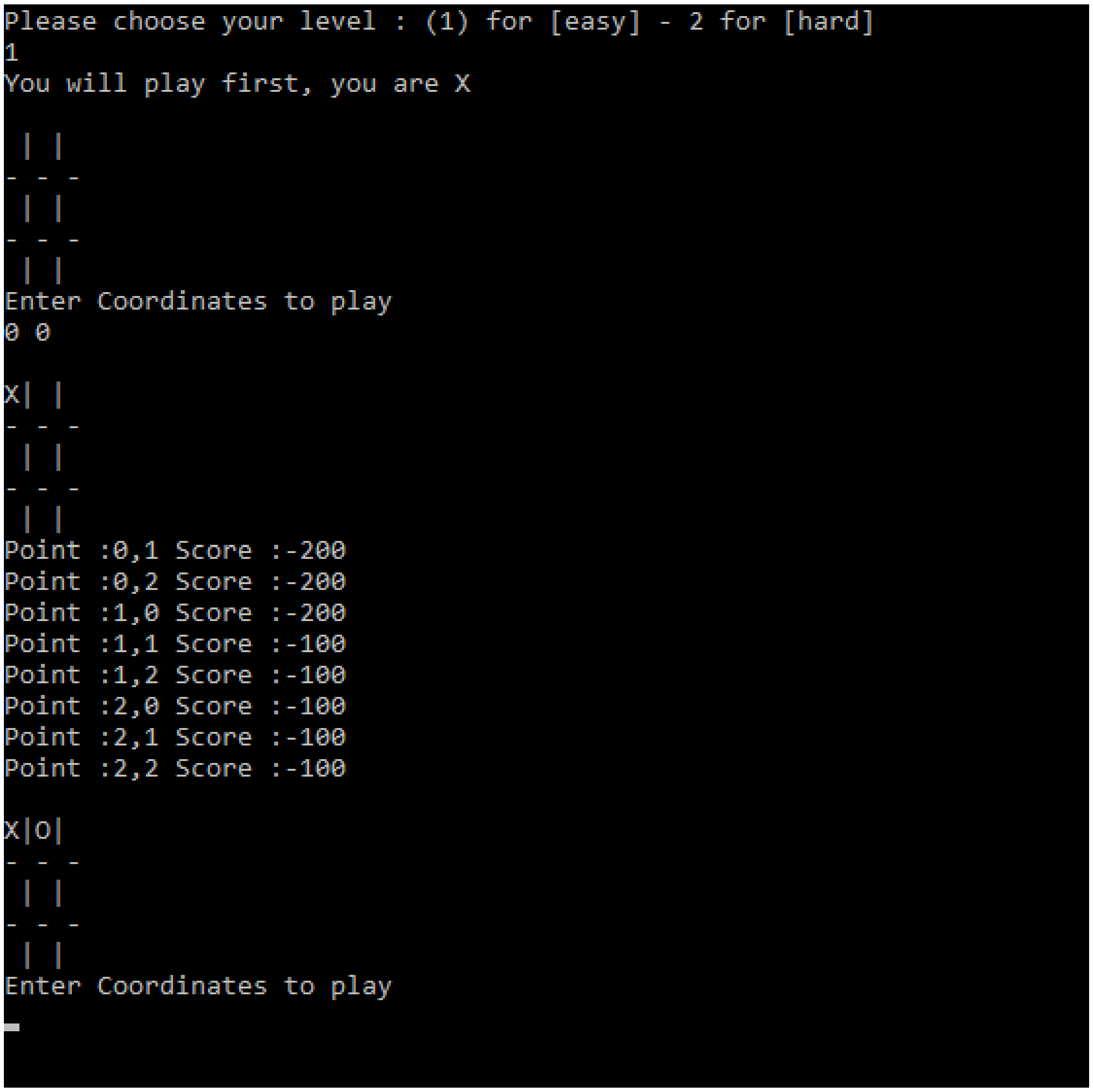
If the player tries to be dump and plays @[0,2] and not to block the computer move, the computer will take the chance and play in the max score box to win which is @[2,1].
And so the computer wins.

Same case but for easy difficulty :

To implement easy difficulty we just get the computer to play the worst move not the best one so he tries to minimize the score also (plays the dumbest move) .

The human will play the same moves as he have done before and se if he can win.

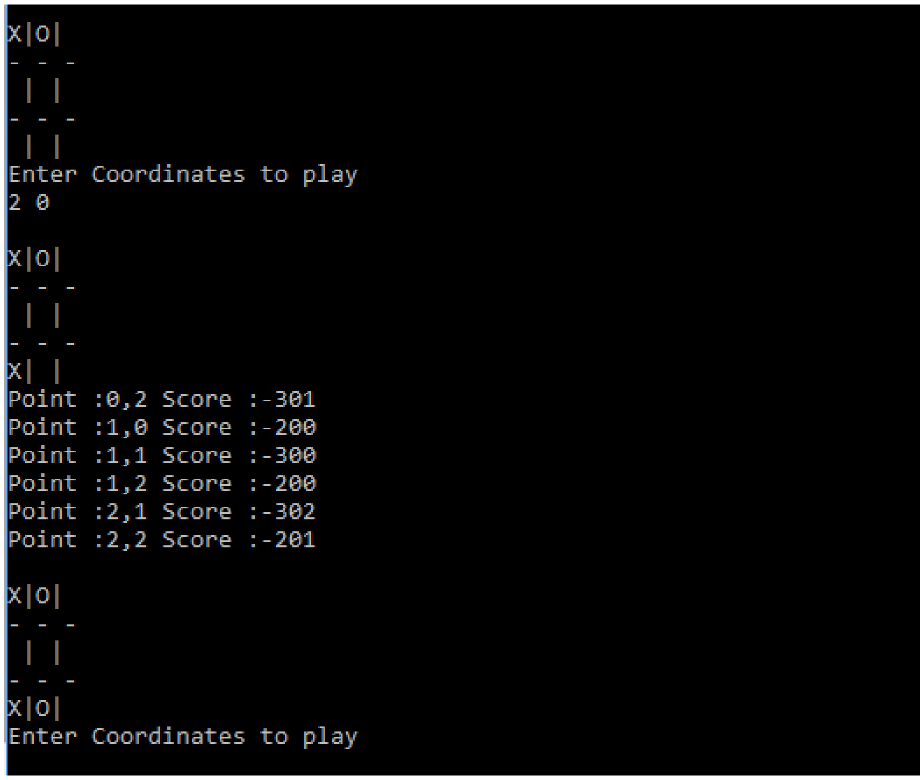
- Human plays @[0,0]



```
Please choose your level : (1) for [easy] - 2 for [hard]
1
You will play first, you are X

 | |
- - -
 | |
- - -
 | |
Enter Coordinates to play
0 0

X| |
- - -
 | |
- - -
 | |
Point :0,1 Score :-200
Point :0,2 Score :-200
Point :1,0 Score :-200
Point :1,1 Score :-100
Point :1,2 Score :-100
Point :2,0 Score :-100
Point :2,1 Score :-100
Point :2,2 Score :-100

X|O|
- - -
 | |
- - -
 | |
Enter Coordinates to play
```

The computer will play @ the dumbest box which is [1,0] which will minimize the score .

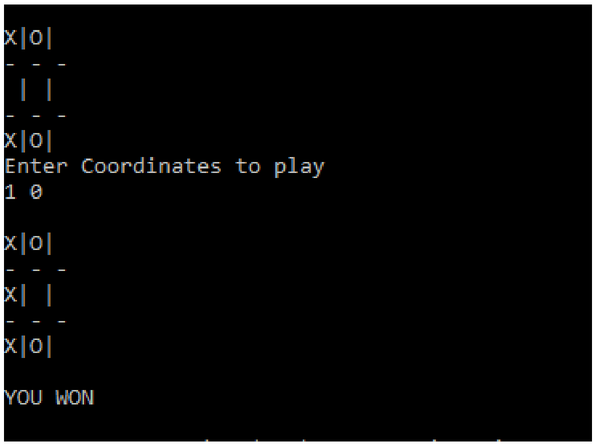
- Human plays @[2-0]



```
X|O|
- - -
 | |
- - -
 | |
Enter Coordinates to play
2 0

X|O|
- - -
 | |
- - -
X| |
Point :0,2 Score :-301
Point :1,0 Score :-200
Point :1,1 Score :-300
Point :1,2 Score :-200
Point :2,1 Score :-302
Point :2,2 Score :-201

X|O|
- - -
 | |
- - -
X|O|
Enter Coordinates to play
```

And so the computer will play @[2,1] and will to even try to block the opponent move

- Human move @[1,0] and so he Won.



```
X|O|
- - -
 | |
- - -
X|O|
Enter Coordinates to play
1 0

X|O|
- - -
X| |
- - -
X|O|

YOU WON
```