Course number: CS440Q4
Members: Haoxuan Sun, Junzhe Wu
Date: 2/25/2019

Section 1:

To solve the CSP, we use the Knuth's algorithm X. Algorithm X can be used to find all solutions to the exact cover problem, and it is a recursive, nondeterministic, depth-first, backtracking algorithm. For example, if we have a matrix A:

	A	B	C	D
P	0	1	0	1
Q	1	0	0	0
R	1	1	0	0
S	0	0	1	0
T	0	0	1	1

In this example, the rows P, Q, S and the rows R,T are the solutions for exact cover problem. The algorithm has 6 steps:

1. If the matrix A has no columns, the current partial solution is a valid solution. Terminate successfully.
2. Choose a column c with the lowest number of 1s.
3. Choose a row r such that $A_{r,c}$=1.
4. Include row r in the partial solution.
5. For each column j such that $A_{r,j}$=1,
   For each row i such that $A_{i,j}$=1,
   Delete row i from matrix A.
   Delete column j from matrix A.
6. Repeat recursively on the reduced matrix A.

So we just need to covert CSP to a matrix,then we can apply the algorithm X to get the solution. Since board has 60 cells, we first have 60 columns in the matrix if all cells are 1s. If we have a piece 1 in the board as below,

(0,0)	(0,1)	(0,2)	(0,3)	(0,4)	(0,5)	(0,6)	(0,7)	(0,8)	(0,9)
(1,0)	(1,1)	(1,2)	(1,3)	(1,4)	(1,5)	(1,6)	(1,7)	(1,8)	(1,9)
(2,0)	(2,1)	(2,2)	(2,3)	(2,4)	(2,5)	(2,6)	(2,7)	(2,8)	(2,9)
(3,0)	(3,1)	(3,2)	(3,3)	(3,4)	(3,5)	(3,6)	(3,7)	(3,8)	(3,9)
(4,0)	(4,1)	(4,2)	(4,3)	(4,4)	(4,5)	(4,6)	(4,7)	(4,8)	(4,9)
(5,0)	(5,1)	(5,2)	(5,3)	(5,4)	(5,5)	(5,6)	(5,7)	(5,8)	(5,9)

We can have a row like below,

	1	2	3	...	12	(0,0)	(0,1)	(0,2)	(0,3)	(0,4)	(0,5)	(0,6)	(0,7)	(0,8)	(0,9)	(1,0)	....
P1(0,0)	1	0	0	0..0	0	1	1	1	1	0	0	0	0	0	0	1	0..0

1,2,3...12 means each piece's index. If all cells are 1s, we will have 12+60=72 columns. If a

cell is 0s, we should not consider it as as a column. Each piece can be rotated and flipped, so a piece has at most 8 different orientations. And each orientations can be placed to 60 coordinates. The number of rows is at most 8*12*60=5760.
Finally, applying algorithm X to get the solution.

Section 2:

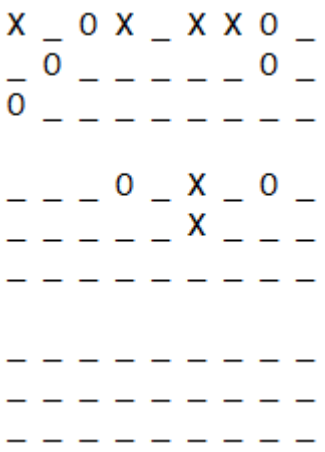
1. Offensive(minimax) vs Defensive(minimax): Max goes first
   Final game board position:

   ```
   0 _ X 0 _ _ 0 X _
   _ X _ _ _ _ _ _ _
   X _ _ _ _ _ _ _ _

   _ _ _ X _ 0 _ _ _
   _ _ _ _ _ _ _ _ _
   _ _ _ _ _ _ _ _ _

   _ _ _ _ _ _ _ _ _
   _ _ _ _ _ _ _ _ _
   _ _ _ _ _ _ _ _ _
   ```

   Expended Nodes: [793, 775, 688, 742, 588, 651, 635, 677, 480]
   Final winner: offensive agent
2. Offensive(minimax) vs Defensive(alpha-beta): Max goes first
   Final game board position:

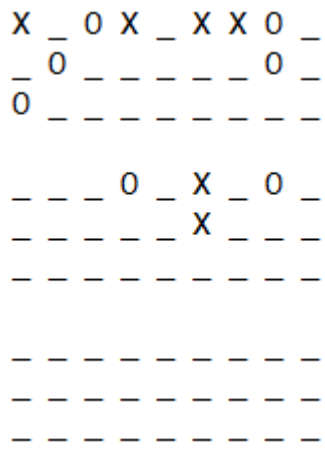
   ```
   0 _ X 0 _ _ 0 X _
   _ X _ _ _ _ _ _ _
   X _ _ _ _ _ _ _ _

   _ _ _ X _ 0 _ _ _
   _ _ _ _ _ _ _ _ _
   _ _ _ _ _ _ _ _ _

   _ _ _ _ _ _ _ _ _
   _ _ _ _ _ _ _ _ _
   _ _ _ _ _ _ _ _ _
   ```

   Expanded Nodes: [793, 201, 688, 213, 588, 262, 635, 179, 480]
   Final winner: offensive agent
3. Offensive(alpha-beta) vs Defensive(minimax): Min goes first
   Final game board position:



Expanded Nodes: [793, 230, 688, 213, 588, 262, 635, 394, 531, 311, 630, 232, 441]
Final winner: defensive agent

4. Offensive(alpha-beta) vs Defensive(alpha-beta): Min goes first
   Final game board position:



Expanded Nodes: [177, 230, 280, 213, 253, 262, 215, 394, 211, 311, 253, 232, 255]
Final winner: defensive agent

Section 3:

offensive agent vs my agent:

1. formulation and advantages of evaluation function
   We take the situation of opposite player into consideration. We set our designed player is the Min player and the score of designed player is minus. So if there is “three in a line” of the Max player, the score of designed player will plus 10000. If there is “two in a line” of the Max player, the score of designed player will plus 460. The rest part is just as same as the offensive player.
   Moreover, in our designed alpha-beta algorithm, we add a judgement condition in the second search level. If the opposite player has the possibility to win in in the second search level, we will return the +10000 value and break the circulation. Since the score for our designed player is minus, so we will never choose +10000 in the first search level.
2. percentage of winning time: 100%
3. number of expanded nodes:

[177, 230, 280, 213, 253, 255, 205, 393, 339, 375, 155] Min goes first, Min wins.
[177, 230, 280, 213, 253, 262, 215, 394, 339, 376, 153] Min goes first, Min wins
[177, 230, 280, 340, 260, 325, 341, 309, 320, 281, 284, 261, 350, 163] Max goes first, Min wins
[177, 232, 334, 334, 312, 246, 379, 189] Max goes first, Min wins
[177, 230, 280, 271, 245, 278, 287, 376, 272, 481, 179] Min goes first, Min wins.
[240, 290, 222, 262, 264, 213, 403, 350, 385, 155] Max goes first, Min wins
[177, 230, 280, 288, 288, 324, 397, 245, 326, 247, 373, 181] Max goes first, Min wins
[240, 290, 351, 203, 260, 263, 271, 376, 187] Min goes first, Min wins
[177, 230, 280, 213, 253, 255, 213, 387, 339, 369, 146] Min goes first, Min wins
[177, 230, 280, 340, 260, 325, 340, 309, 319, 279, 346, 303, 421, 221] Max goes first, Min wins
[177, 230, 280, 340, 260, 325, 341, 309, 320, 293, 324, 48] Max goes first, Min wins
[177, 232, 334, 224, 284, 248, 297, 330, 378, 370, 154] Min goes first, Min wins.
[177, 230, 280, 340, 260, 325, 306, 217, 306, 231, 424, 175] Max goes first, Min wins
[177, 230, 280, 213, 253, 255, 205, 393, 339, 375, 155] Min goes first, Min wins.
[177, 230, 280, 340, 260, 264, 380, 245, 319, 240, 366, 181] Max goes first, Min wins
[177, 230, 280, 213, 253, 255, 205, 393, 339, 318, 154] Min goes first, Min wins.
[177, 230, 280, 340, 260, 325, 341, 187, 319, 240, 366, 183] Max goes first, Min wins
[240, 290, 222, 262, 264, 213, 403, 350, 385, 155] Max goes first, Min wins
[177, 230, 280, 340, 260, 325, 341, 309, 320, 293, 324, 48] Max goes first, Min wins
[240, 290, 351, 203, 260, 263, 271, 376, 187] Min goes first, Min wins

4. Final game board position:

```
X _ 0 _ _ X X 0 _
_ 0 _ _ _ _ _ 0 _
_ _ _ _ _ _ _ 0 _

_ _ _ _ _ X 0 _ _
_ _ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _ _

_ _ _ _ _ X _ _ _
_ _ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _ _
```

Startidx=5, Min goes first. Minplayer wins.
Movelist:[(3, 6), (0, 0), (0, 2), (0, 6), (1, 1), (3, 5), (0, 7), (0, 5), (2, 7), (6, 5), (1, 7)]

X _ 0 _ _ X X 0 _
_ 0 _ _ _ _ _ 0 _
_ _ _ _ _ _ _ 0 _

_ _ _ _ _ X _ _ _
_ _ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _ _

0 _ _ _ _ X _ _ _
_ _ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _ _

Startidx=4, Min goes first. Minplayer wins.
Movelist: [(3, 3), (0, 0), (0, 2), (0, 6), (1, 1), (3, 5), (0, 7), (0, 5), (2, 7), (6, 5), (1, 7)]

0 _ X X 0 _ _ _ 0
0 _ _ _ _ _ _ _ _
0 _ _ _ _ _ X _ _

X _ _ _ _ _ X _ _
_ _ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _ _

_ _ _ _ _ _ _ X _
_ _ _ _ _ _ _ _ _
0 _ X _ _ _ _ _ 0

Startidx=5, Max goes first. Minplayer wins.
Movelist: [(3, 6), (0, 0), (0, 2), (0, 8), (2, 6), (8, 0), (8, 2), (8, 8), (6, 7), (0, 4), (0, 3), (1, 0), (3, 0), (2, 0)]

0 _ X X _ _ X _ 0
0 _ _ _ _ _ _ _ _
0 _ _ _ _ _ _ _ _

X _ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _ _

_ _ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _ _

Startidx=1, Max goes first. Minplayer wins.
Movelist: [(0, 3), (0, 0), (0, 2), (0, 8), (0, 6), (1, 0), (3, 0), (2, 0)]

Section 4:

1. percentage of winning time for our agent: 60%
2. advantage: it will stop me to get “two in line” and “three in line ” in the three depth search since this evaluation function takes the situation of opposite player into consideration.

   Disadvantages: the agent only has 3 depths which means it can not look too far through the game. The values we set for each condition need to be adjusted. We should find the suitable values based on multiple games.
3. final game boards position:

```
X _ 0 0 X _ _ X X
X _ 0 0 _ _ 0 _ _
_ _ 0 _ _ _ 0 _ _

X _ _ _ _ 0 0 _ _
X 0 X _ X 0 0 _ _
0 X _ _ _ _ _ X X

X _ _ _ _ X _ 0 _
_ _ _ 0 X _ _ _ _
X _ 0 _ 0 _ _ 0 X
```

Start id=5, agent(Min) goes first. Min wins.

Move List: [(3, 6), (0, 0), (0, 2), (0, 8), (2, 6), (8, 0), (8, 2), (8, 8), (6, 7), (0, 4), (1, 3), (4, 0), (4, 1), (4, 4), (4, 5), (5, 7), (8, 4), (7, 4), (3, 5), (0, 7), (0, 3), (1, 0), (5, 0), (6, 0), (1, 2), (5, 8), (8, 7), (6, 5), (1, 6), (5, 1), (7, 3), (4, 2), (4, 6), (3, 0), (2, 2)]

```
X _ _ _ X _ 0 _ _
_ X _ 0 0 _ _ X _
0 _ _ _ X _ 0 _ _

_ _ _ 0 0 X _ _ 0
X 0 _ _ _ X _ _ 0
_ _ _ _ _ _ X _ 0

_ _ _ _ 0 X _ X _
_ _ X _ _ X _ _ _
_ 0 X _ 0 _ _ X 0
```

Start id=2, human(Max) goes first. Min wins.

Move list: [(1, 7), (3, 3), (0, 0), (2, 0), (8, 2), (8, 8), (8, 7), (6, 4), (0, 4), (1, 3), (4, 0), (4, 1), (4, 5), (4, 8), (5, 6), (8, 1), (6, 5), (0, 6), (1, 1), (3, 4), (2, 4), (8, 4), (7, 5), (5, 8), (6, 7), (1, 4), (3, 5), (2, 6), (7, 2), (3, 8)]

```
O X X X O _ _ _ X
O X _ X _ _ O _ X
O _ _ _ O _ O _ _

O _ _ O _ O X X _
O _ X _ _ _ _ O O
_ _ X X _ _ _ _ _

_ _ _ _ _ _ _ O O
_ _ O X X _ _ _ _
X X O _ O O _ X X
```

Start id=0, agent(Min) goes first. Min wins.

Move list: [(0, 0), (1, 1), (3, 5), (0, 8), (2, 6), (8, 1), (8, 5), (8, 8), (6, 7), (1, 3), (4, 0), (4, 2), (4, 7), (5, 3), (7, 2), (3, 7), (2, 4), (7, 4), (3, 3), (0, 2), (1, 6), (5, 2), (6, 8), (1, 8), (4, 8), (3, 6), (2, 0), (8, 0), (8, 2), (8, 7), (8, 4), (7, 3), (3, 0), (0, 1), (0, 4), (0, 3), (1, 0)]

```
X _ O _ X _ O _ X
X _ _ O _ _ _ _ X
X _ O O X _ O _ _

_ O _ _ X _ O _ _
X O _ _ X O O _ X
X _ _ _ _ O _ _ X

X _ _ O _ O _ O X
_ O O _ X _ _ _ _
X _ O _ X _ X O X
```

Start id=7, human(Max) goes first. Max wins.

Move list:[(8, 4), (6, 3), (0, 0), (0, 2), (0, 8), (2, 6), (8, 0), (8, 2), (8, 8), (6, 7), (0, 4), (1, 3), (4, 0), (4, 1), (4, 4), (4, 5), (4, 8), (3, 6), (1, 0), (3, 1), (2, 4), (6, 5), (1, 8), (4, 6), (5, 0), (7, 1), (3, 4), (2, 3), (6, 0), (2, 2), (8, 6), (7, 2), (5, 8), (8, 7), (7, 4), (5, 5), (6, 8), (0, 6), (2, 0)]

Section 5:

1. part1's code and report are written by Haoxuan Sun
2. part2's code and report are written by Junzhe WU