Istanbul Technical University
Faculty of Computer and Informatics
Computer Engineering Department

BLG 435E
AI
Homework II

Uğur Uysal - 150140012

Dec 9^{th}, 2018

Contents

1 Question I **1**

1.1 Model 1

1.2 Implementation and Algorithm 1

2 Question II **3**

2.1 Part I 3

2.2 Part II 3

3 Question III **6**

3.1 Part I 6

3.2 Part II 7

1 Question I

1.1 Model

I used blocks as variables and number of blocks as a domain. For example for the given first example the list of variables are these. The origin is top-left corner and it‘s coordinates increasing from left to right and from top to bottom.

Blocks represented as the direction and starting point. For vertical blocks. starting points is top, and for horizontal blocks, it is left.

- HORIZONTAL 0 3
- HORIZONTAL 1 1
- HORIZONTAL 1 4
- HORIZONTAL 2 2
- HORIZONTAL 2 5
- HORIZONTAL 5 0
- VERTICAL 2 1
- VERTICAL 3 3
- VERTICAL 3 6

Domain is values that can variables take. I used numbers from 0 to number of variables and I assigned the values as which order the block must be placed in grid.

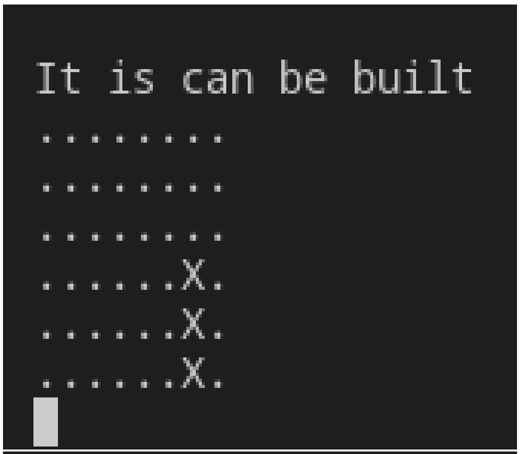
This is the one of the solutions. My algorithms found that.

1. VERTICAL 3 6
2. VERTICAL 3 3
3. HORIZONTAL 5 0
4. VERTICAL 2 1
5. HORIZONTAL 2 5
6. HORIZONTAL 2 2
7. HORIZONTAL 1 4
8. HORIZONTAL 1 1
9. HORIZONTAL 0 3

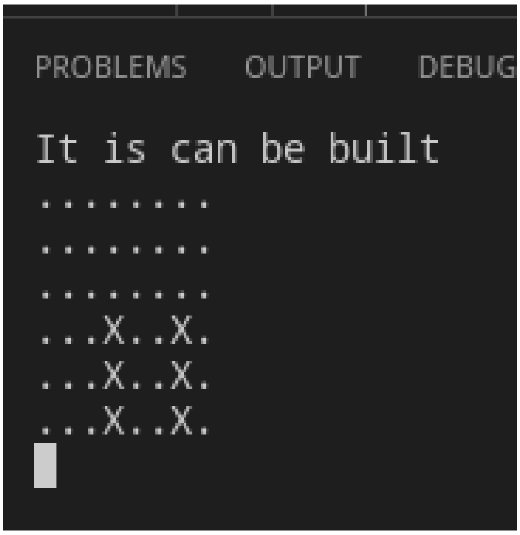
The constraints are implemented as given in PDF. Also no blocks can be placed in same time, i.e their value can not be same.

1.2 Implementation and Algorithm

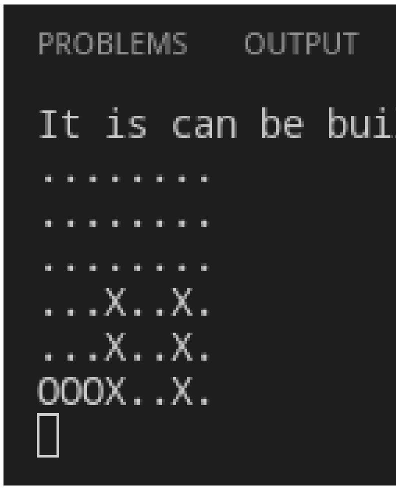
The algorithm does not stuck. Because if there is not any available assignment in domain space it just stops. Example output is given with visualization at the next page.



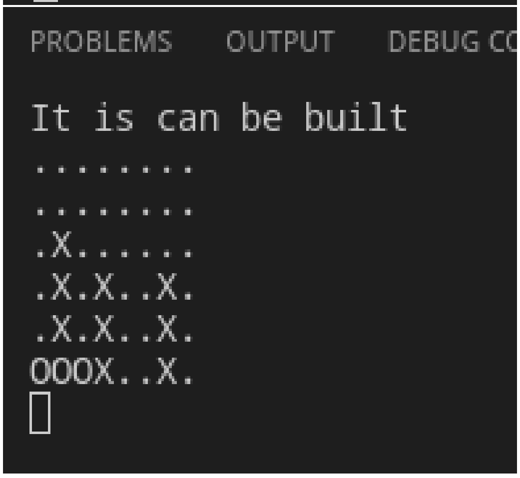
```
It is can be built
........
........
........
......X.
......X.
......X.
```



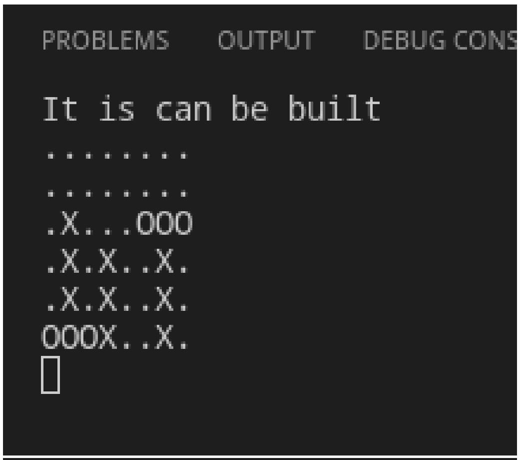
```
PROBLEMS    OUTPUT    DEBUG
It is can be built
........
........
........
...X..X.
...X..X.
...X..X.
```



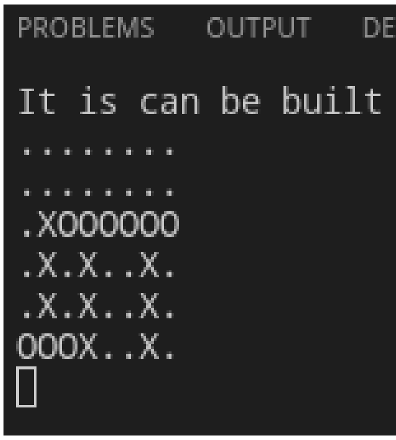
```
PROBLEMS    OUTPUT
It is can be bui
........
........
........
...X..X.
...X..X.
OOOX..X.
```



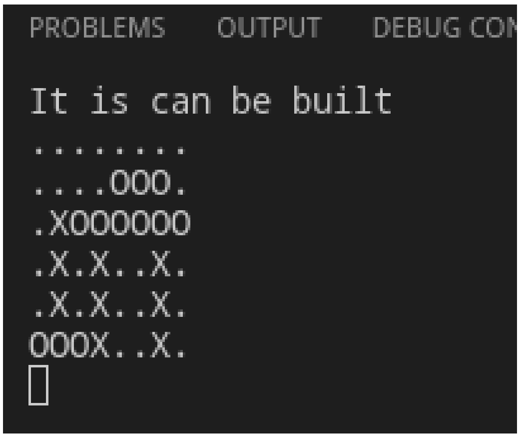
```
PROBLEMS    OUTPUT    DEBUG CO
It is can be built
........
........
.X......
.X.X..X.
.X.X..X.
OOOX..X.
```



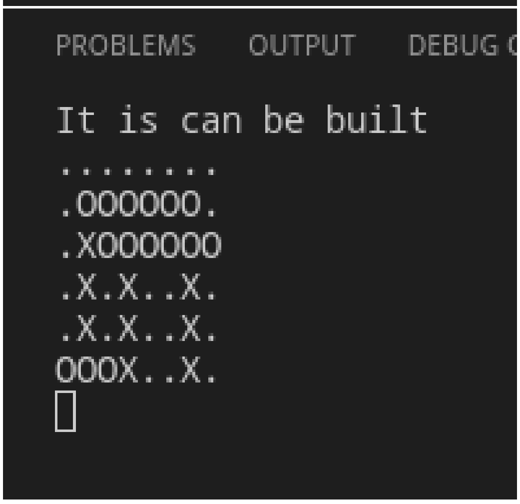
```
PROBLEMS    OUTPUT    DEBUG CONS
It is can be built
........
........
.X...OOO
.X.X..X.
.X.X..X.
OOOX..X.
```



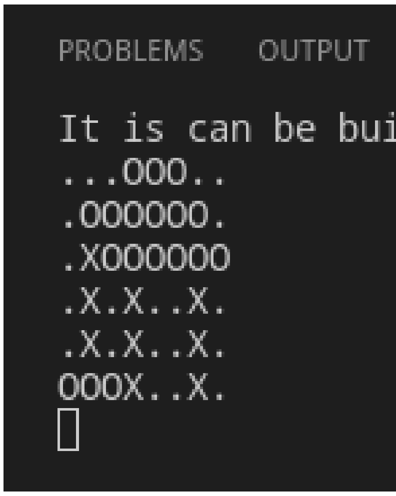
```
PROBLEMS    OUTPUT    DE
It is can be built
........
........
.XOOOOOO
.X.X..X.
.X.X..X.
OOOX..X.
```



```
PROBLEMS    OUTPUT    DEBUG CON
It is can be built
........
....OOO.
.XOOOOOO
.X.X..X.
.X.X..X.
OOOX..X.
```



```
PROBLEMS    OUTPUT    DEBUG C
It is can be built
........
.OOOOOO.
.XOOOOOO
.X.X..X.
.X.X..X.
OOOX..X.
```

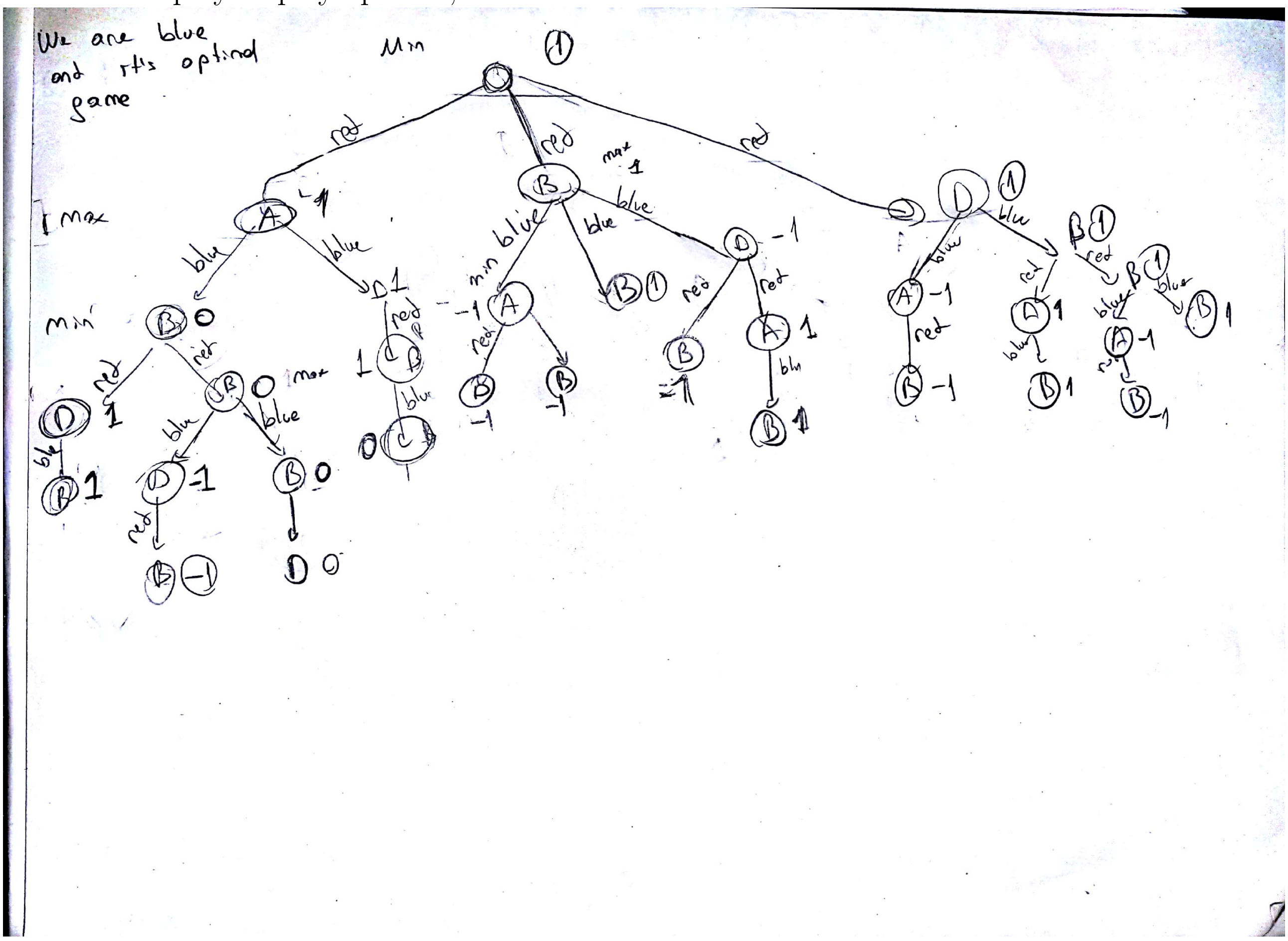


```
PROBLEMS    OUTPUT
It is can be bui
...OOO..
.OOOOOO.
.XOOOOOO
.X.X..X.
.X.X..X.
OOOX..X.
```

2 Question II

2.1 Part I

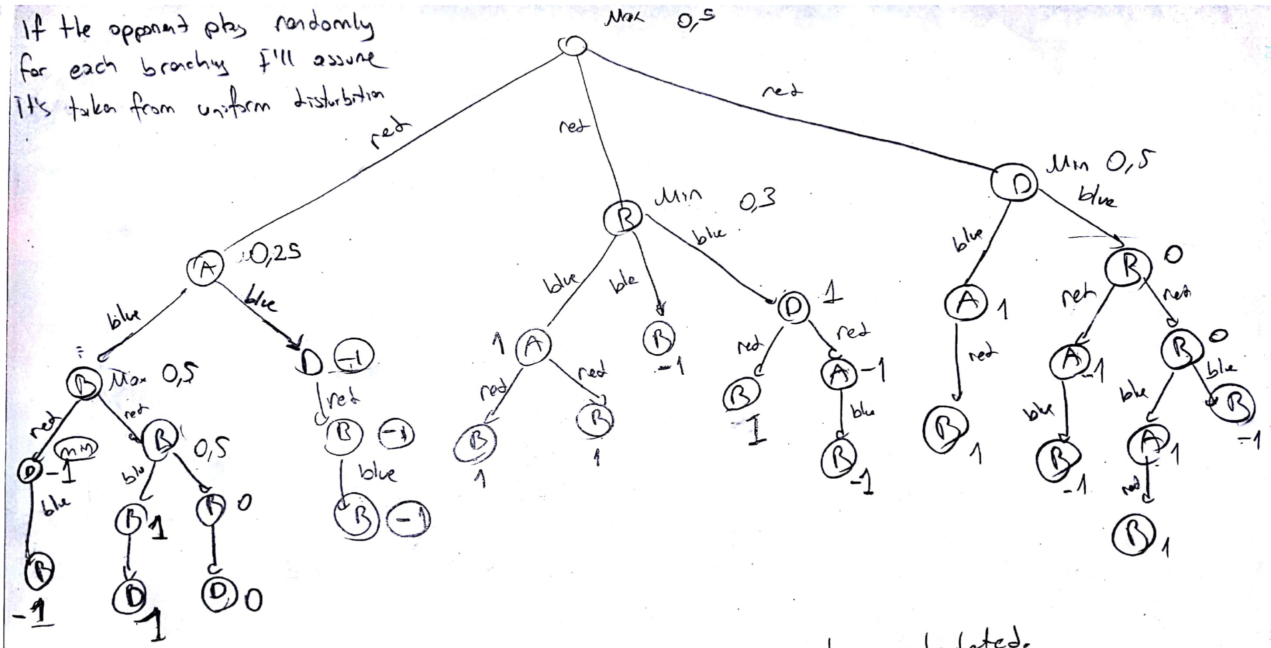
This is the mini-max tree. The given state of the board Red moves first and we can obverse that if both players play optimal, Blue win.



2.2 Part II

This is the mini-max tree. The given state of the board Red moves first. Calculations are made based on expected value since blue plays randomly.

If the opponent plays randomly for each branching I'll assume it's taken from uniform distribution



For each branching of blue player expected value calculated.

3 Question III

3.1 Part I

Knowledge Base

$\forall x, \exists y\ [student(x) \wedge StudentClub(y)] \Rightarrow Attends(x,y)$

$\forall x\ [Student(x) \wedge Attends(x, LIT\ CLUB)] \Rightarrow \neg Like(x, COLA)$

$\forall x\ [Student(x) \wedge Attend(x, CINEMA\ CLUB)] \Rightarrow Like(x, POPCORN)$

- $Like(Ayse, COLA)$, $\neg Like(Ayse, POPCORN)$
- $Like(BARIS, COLA)$, $Like(BARIS, COLA)$
- $\forall x [\ \neg Likes(Ayse, x)\ ] \Rightarrow Like(Cem, x)$.

$Student(x) \rightarrow$ true if x is student
$StudentClub(x) \rightarrow$ true if x is studentClub
$Attends(x,y) \rightarrow$ true if x attends to club y
$Like(x,y) \rightarrow$ true if x likes y.

(Part II).

For Ayse: Proof by Contra.

(1) $KB \wedge \neg Attends(Ayse, Game\ Club)$.

$\forall x, \exists y\ [student(x) \wedge StudentClub(y))] \Rightarrow Attend(x,y) \Rightarrow \left(\begin{array}{l} Attends(Ayse, LIT\ CLB) \vee \\ Attend(Ayse, Game\ Club) \vee \\ Attend(Ayse, CineCl.) \end{array} \right) \rightarrow Q$

$\neg [Student(x) \wedge StudentClub(y) \Rightarrow Attend(x,y)] \vee Q$

$\neg [\neg (student(x) \wedge StudentClub(y)) \vee Attends(x,y)] \vee Q$

$((student(x) \wedge studentClub(y)) \wedge \neg Attends(x,y)) \vee Q$

$\rightarrow$ In KB

$\neg Attends(Ayse, Game\ Club)$

$Attends(Ayse, LitClub) \vee Attends(Ayse, Cinema\ Club)$

$\rightarrow$ $Student(Ayse) \wedge Attends(Ayse, Lit\ Club) \Rightarrow \neg Like(Ayse, COLA)$ ($Like(Ayse, COLA)$ in KB
($\vee$ operation leads to false

we infer $\neg Attends(Ayse, Lit\ Club)$,

$student(Ayse) \wedge Attends(Ayse, CinemaClub) \Rightarrow Like(Ayse, POPCORN)$ ($\neg Like(Ayse, Cola)$
in KB
$\vee$ leads to fal.

we infer $\neg Attends(Ayse, CinemaClub)$

3.2 Part II

Now we have contradict in (1).

so KB∧¬Attends(Ayse, Game Club) is unsatisfable.

Ayse Attends to Game Club.

F Boris:

KB ⇒ ¬Attends(Boris, Game Club)

Student(Boris) ⇒ Attends(Boris, Game Club) ∨ Attends(Boris, LIT) ∨ Attend(Boris, CINEMA)

Attends(Boris, LIT CLUB) ⇒ ¬Like(Boris, CoLA)

¬Attends(Boris, LIT CLUB) ∨ ¬Like(Boris COLA) ∨ Like(BARIS, COLA) so

Boris does oth Lit CLR

Attends(BARIS, CIMEA CLB) ⇒ Like(BARIS, POPCORN)

¬Attends(BARIS, CINEMA CLB) ∨ Like(BARIS, POPCORN) ∨ Like(BARIS, POPCORN)

We cannot infer whether boris attends CINEM CLUB or Game CLUB,

For Cem:

KB ⇒ ¬Attends(Cem, Game Club)

Student(Cem) ⇒ Attends(Cem, LIT) ∨ Attends(Cem, Cinema) ∨ Attends(Cem, Game).

¬Like(Ayse, POPCORN) ⇒ Like(Cem, POPCORN)

Attends(Cem, Cinema) ⇒ Like(CEM, POPCORN)

So we cannot infer which Club Cem attends.

it could be any club.