Introduction to AI Programming Assignment #3 Report

Student ID: **Name:**

Code interpretation:

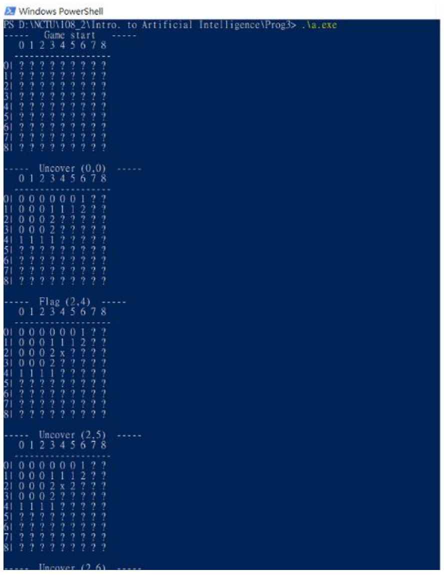
The program is in two parts, Game Master module and Player module. Game master module is implemented by class ***Minesweeper***. In this class, there has 2 private objects, ***gField*** is a 2-dimension integer array that has complete information of every grids, an integer ***unsolved*** is the number of grids that player should claim it is safe.
In the Prog3.pdf, player reaches goal if KB's size is 0. But in the real game, goal test is the job of Game Master side. Thus, once Game Master detects that ***unsolved*** is zero or player clicks a grid that has mine, the program will be terminated directly.
Player module is in main function. The process is as same as the Game flow section in Prog3.pdf. The only different is that I do some change in generating clause from hint. Because during the matching, we only match the clause which has less than 2 literals. In the case m>n>0, it isn't necessary to generate the clause which has more than 2 literals. These clauses can't use during matching but they can make the time of matching become longer. The case m>n>0 will generate positive clause if m-n=1, generate negative clause if n=1, drop the clause if number of literals > 2. The clause being drop will be generate after more information is known.

Design of KB (Knowledge Base):

The way of designing KB is one of the main problems in the program assignment. My KB is implemented by set<vector<int> >. The reason why I use set because dealing with duplicate is annoying. The subsumption is a case that I don't need to deal with because I only save the 2-literals clause in KB. Every vector<int> is a clause. Integer in the vector is the literal. Every literal has 5 digits. The first digit is 1 means that this literal is positive, is 0 means it is negative. The second and third digit is the x coordinate. The forth and last digit is the y coordinate. In this design literal L1 can eliminate literal L2 if |L1 – L2| = 10000 which makes the matching easily. I don't design KB0 by myself because I think the field player can see is already give player enough knowledge to determine the situation.

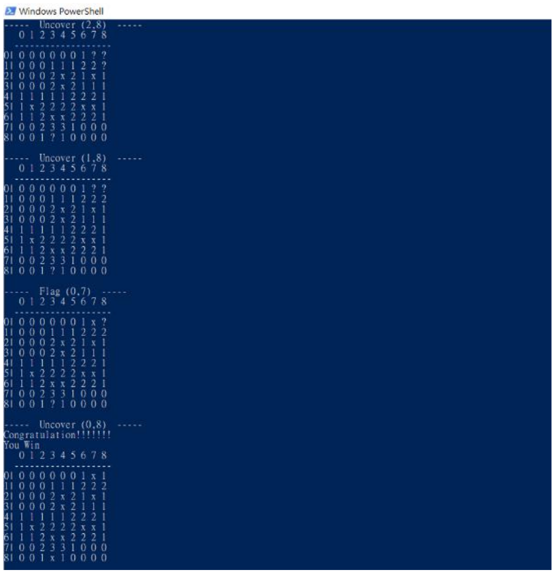
Testing result:

level 1 (9x9 with 10 mines):



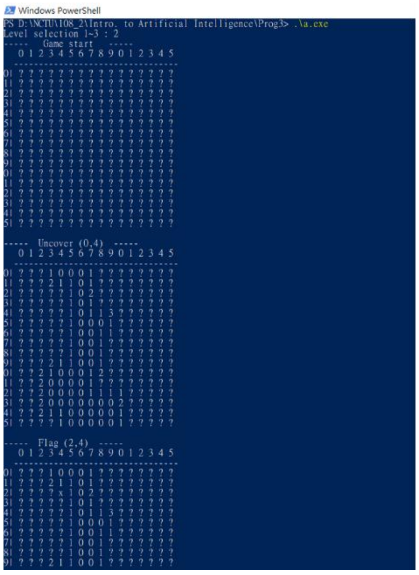
```
PS D:\NCTU\108_2\Intro. to Artificial Intelligence\Prog3> .\a.exe
-----   Game start   -----
   0 1 2 3 4 5 6 7 8
  ------------------
0| ? ? ? ? ? ? ? ? ?
1| ? ? ? ? ? ? ? ? ?
2| ? ? ? ? ? ? ? ? ?
3| ? ? ? ? ? ? ? ? ?
4| ? ? ? ? ? ? ? ? ?
5| ? ? ? ? ? ? ? ? ?
6| ? ? ? ? ? ? ? ? ?
7| ? ? ? ? ? ? ? ? ?
8| ? ? ? ? ? ? ? ? ?

-----  Uncover (0,0)  -----
   0 1 2 3 4 5 6 7 8
  ------------------
0| 0 0 0 0 0 0 1 ? ?
1| 0 0 0 1 1 1 2 ? ?
2| 0 0 0 2 ? ? ? ? ?
3| 0 0 0 2 ? ? ? ? ?
4| 1 1 1 1 ? ? ? ? ?
5| ? ? ? ? ? ? ? ? ?
6| ? ? ? ? ? ? ? ? ?
7| ? ? ? ? ? ? ? ? ?
8| ? ? ? ? ? ? ? ? ?

-----  Flag (2,4)  -----
   0 1 2 3 4 5 6 7 8
  ------------------
0| 0 0 0 0 0 0 1 ? ?
1| 0 0 0 1 1 1 2 ? ?
2| 0 0 0 2 x ? ? ? ?
3| 0 0 0 2 ? ? ? ? ?
4| 1 1 1 1 ? ? ? ? ?
5| ? ? ? ? ? ? ? ? ?
6| ? ? ? ? ? ? ? ? ?
7| ? ? ? ? ? ? ? ? ?
8| ? ? ? ? ? ? ? ? ?

-----  Uncover (2,5)  -----
   0 1 2 3 4 5 6 7 8
  ------------------
0| 0 0 0 0 0 0 1 ? ?
1| 0 0 0 1 1 1 2 ? ?
2| 0 0 0 2 x 2 ? ? ?
3| 0 0 0 2 ? ? ? ? ?
4| 1 1 1 1 ? ? ? ? ?
5| ? ? ? ? ? ? ? ? ?
6| ? ? ? ? ? ? ? ? ?
7| ? ? ? ? ? ? ? ? ?
8| ? ? ? ? ? ? ? ? ?

-----  Uncover (2,6)  -----
```

●●●



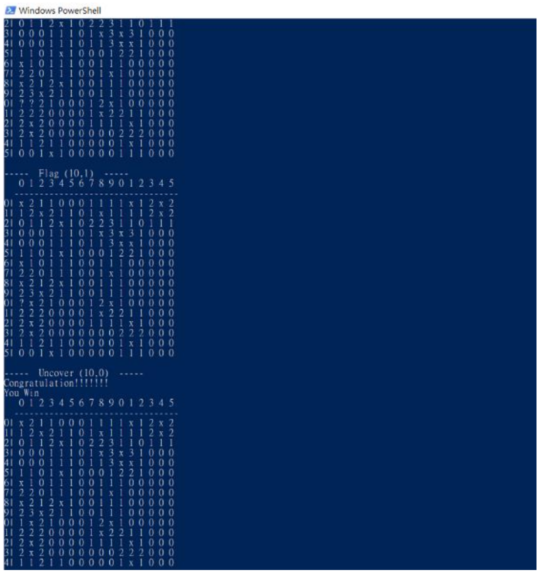
```
-----  Uncover (2,8)  -----
   0 1 2 3 4 5 6 7 8
  ------------------
0| 0 0 0 0 0 0 1 ? ?
1| 0 0 0 1 1 1 2 2 ?
2| 0 0 0 2 x 2 1 x 1
3| 0 0 0 2 x 2 1 1 1
4| 1 1 1 1 1 2 2 2 1
5| 1 x 2 2 2 2 x x 1
6| 1 1 2 x x 2 2 2 1
7| 0 0 2 3 3 1 0 0 0
8| 0 0 1 ? 1 0 0 0 0

-----  Uncover (1,8)  -----
   0 1 2 3 4 5 6 7 8
  ------------------
0| 0 0 0 0 0 0 1 ? ?
1| 0 0 0 1 1 1 2 2 2
2| 0 0 0 2 x 2 1 x 1
3| 0 0 0 2 x 2 1 1 1
4| 1 1 1 1 1 2 2 2 1
5| 1 x 2 2 2 2 x x 1
6| 1 1 2 x x 2 2 2 1
7| 0 0 2 3 3 1 0 0 0
8| 0 0 1 ? 1 0 0 0 0

-----  Flag (0,7)  -----
   0 1 2 3 4 5 6 7 8
  ------------------
0| 0 0 0 0 0 0 1 x ?
1| 0 0 0 1 1 1 2 2 2
2| 0 0 0 2 x 2 1 x 1
3| 0 0 0 2 x 2 1 1 1
4| 1 1 1 1 1 2 2 2 1
5| 1 x 2 2 2 2 x x 1
6| 1 1 2 x x 2 2 2 1
7| 0 0 2 3 3 1 0 0 0
8| 0 0 1 ? 1 0 0 0 0

-----  Uncover (0,8)  -----
Congratulation!!!!!!!
You Win
   0 1 2 3 4 5 6 7 8
  ------------------
0| 0 0 0 0 0 0 1 x 1
1| 0 0 0 1 1 1 2 2 2
2| 0 0 0 2 x 2 1 x 1
3| 0 0 0 2 x 2 1 1 1
4| 1 1 1 1 1 2 2 2 1
5| 1 x 2 2 2 2 x x 1
6| 1 1 2 x x 2 2 2 1
7| 0 0 2 3 3 1 0 0 0
8| 0 0 1 x 1 0 0 0 0
```

level 2 (16x16 with 25 mines):



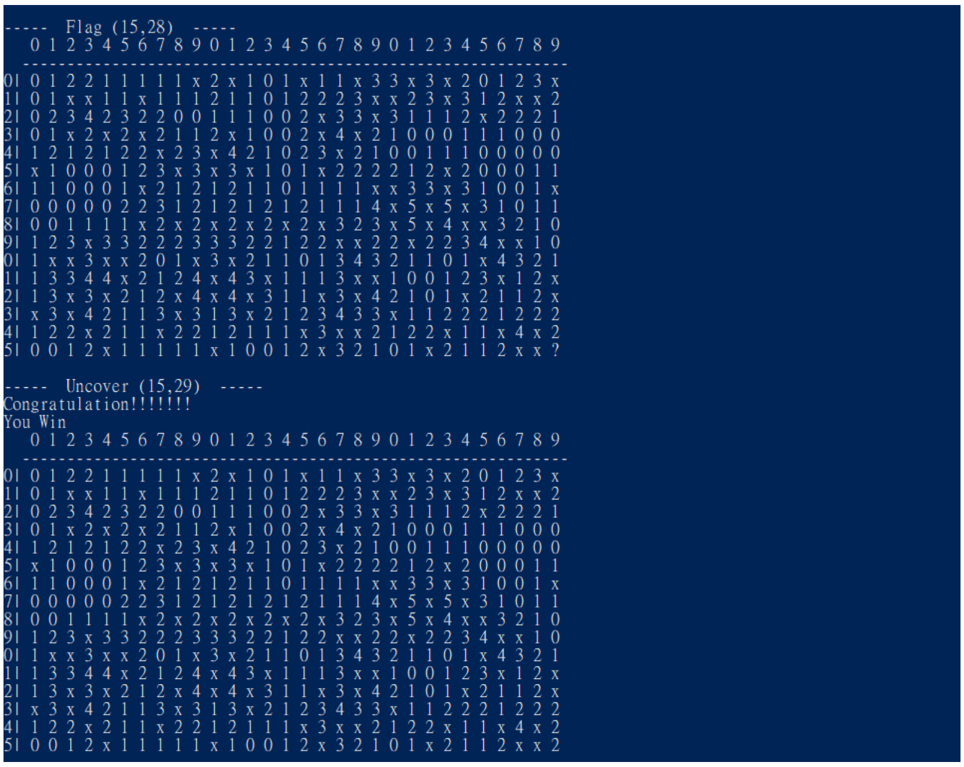
```
PS D:\NCTU\108_2\Intro. to Artificial Intelligence\Prog3> .\a.exe
Level selection 1~3 : 2
-----   Game start   -----
   0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
  --------------------------------
0| ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ?
1| ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ?
2| ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ?
3| ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ?
4| ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ?
5| ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ?
6| ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ?
7| ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ?
8| ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ?
9| ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ?
0| ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ?
1| ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ?
2| ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ?
3| ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ?
4| ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ?
5| ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ?

-----  Uncover (0,4)  -----
   0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
  --------------------------------
0| ? ? ? 1 0 0 0 1 ? ? ? ? ? ? ? ?
1| ? ? ? 2 1 1 0 1 ? ? ? ? ? ? ? ?
2| ? ? ? ? ? 1 0 2 ? ? ? ? ? ? ? ?
3| ? ? ? ? ? 1 0 1 ? ? ? ? ? ? ? ?
4| ? ? ? ? ? 1 0 1 1 3 ? ? ? ? ? ?
5| ? ? ? ? ? 1 0 0 0 1 ? ? ? ? ? ?
6| ? ? ? ? ? 1 0 0 1 1 ? ? ? ? ? ?
7| ? ? ? ? ? 1 0 0 1 ? ? ? ? ? ? ?
8| ? ? ? ? ? 1 0 0 1 ? ? ? ? ? ? ?
9| ? ? ? 2 1 1 0 0 1 ? ? ? ? ? ? ?
0| ? ? 2 1 0 0 0 1 2 ? ? ? ? ? ? ?
1| ? ? 2 0 0 0 0 1 ? ? ? ? ? ? ? ?
2| ? ? 2 0 0 0 0 1 1 1 1 ? ? ? ? ?
3| ? ? 2 0 0 0 0 0 0 0 2 ? ? ? ? ?
4| ? ? 2 1 1 0 0 0 0 0 1 ? ? ? ? ?
5| ? ? ? ? 1 0 0 0 0 0 1 ? ? ? ? ?

-----  Flag (2,4)  -----
   0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
  --------------------------------
0| ? ? ? 1 0 0 0 1 ? ? ? ? ? ? ? ?
1| ? ? ? 2 1 1 0 1 ? ? ? ? ? ? ? ?
2| ? ? ? ? x 1 0 2 ? ? ? ? ? ? ? ?
3| ? ? ? ? ? 1 0 1 ? ? ? ? ? ? ? ?
4| ? ? ? ? ? 1 0 1 1 3 ? ? ? ? ? ?
5| ? ? ? ? ? 1 0 0 0 1 ? ? ? ? ? ?
6| ? ? ? ? ? 1 0 0 1 1 ? ? ? ? ? ?
7| ? ? ? ? ? 1 0 0 1 ? ? ? ? ? ? ?
8| ? ? ? ? ? 1 0 0 1 ? ? ? ? ? ? ?
9| ? ? ? 2 1 1 0 0 1 ? ? ? ? ? ? ?
```

●●●



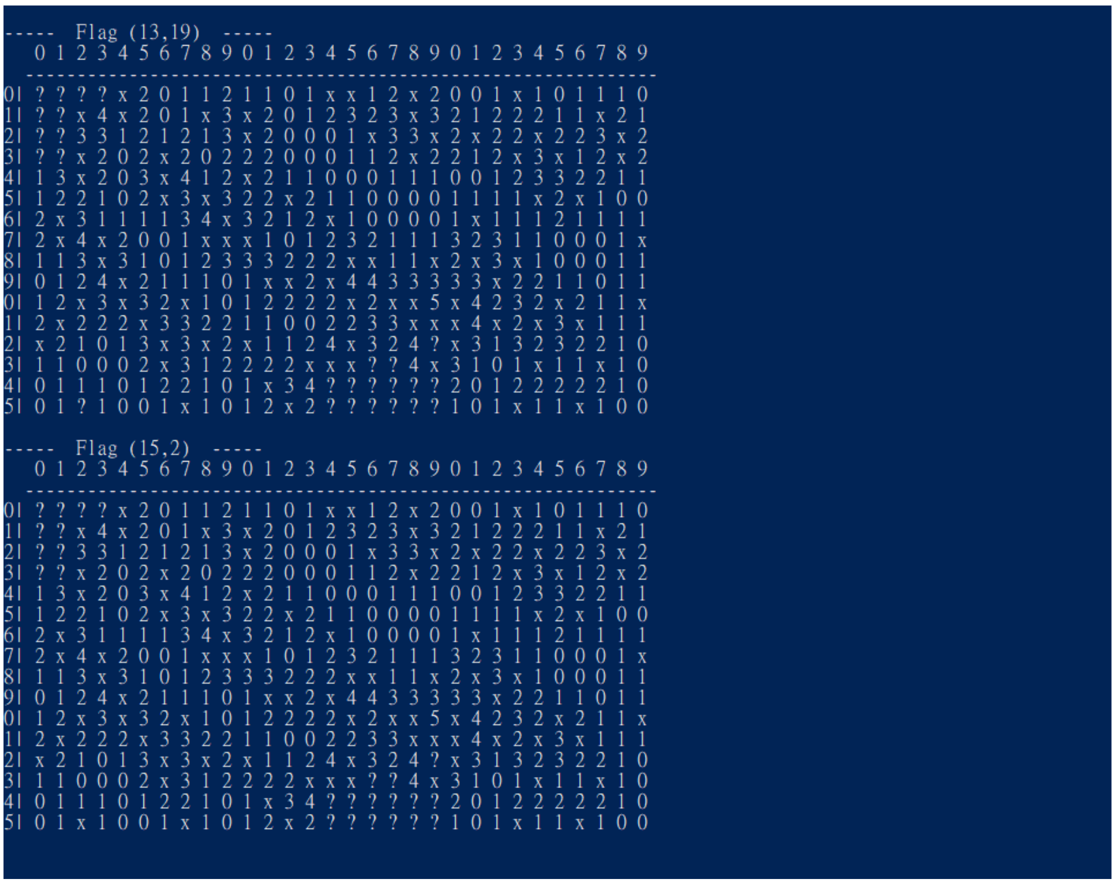
```
2| 0 1 1 2 x 1 0 2 2 3 1 1 0 1 1 1
3| 0 0 0 1 1 1 0 1 x 3 x 3 1 0 0 0
4| 0 0 0 1 1 1 0 1 1 3 x x 1 0 0 0
5| 1 1 0 1 x 1 0 0 0 1 2 2 1 0 0 0
6| x 1 0 1 1 1 0 0 1 1 1 0 0 0 0 0
7| 2 2 0 1 1 1 0 0 1 x 1 0 0 0 0 0
8| x 2 1 2 x 1 0 0 1 1 1 0 0 0 0 0
9| 2 3 x 2 1 1 0 0 1 1 1 0 0 0 0 0
0| ? ? 2 1 0 0 0 1 2 x 1 0 0 0 0 0
1| 2 2 2 0 0 0 0 1 x 2 2 1 1 0 0 0
2| 2 x 2 0 0 0 0 1 1 1 1 x 1 0 0 0
3| 2 x 2 0 0 0 0 0 0 0 2 2 2 0 0 0
4| 1 1 2 1 1 0 0 0 0 0 1 x 1 0 0 0
5| 0 0 1 x 1 0 0 0 0 0 1 1 1 0 0 0

-----  Flag (10,1)  -----
   0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
  --------------------------------
0| x 2 1 1 0 0 0 1 1 1 1 x 1 2 x 2
1| 1 2 x 2 1 1 0 1 x 1 1 1 1 2 x 2
2| 0 1 1 2 x 1 0 2 2 3 1 1 0 1 1 1
3| 0 0 0 1 1 1 0 1 x 3 x 3 1 0 0 0
4| 0 0 0 1 1 1 0 1 1 3 x x 1 0 0 0
5| 1 1 0 1 x 1 0 0 0 1 2 2 1 0 0 0
6| x 1 0 1 1 1 0 0 1 1 1 0 0 0 0 0
7| 2 2 0 1 1 1 0 0 1 x 1 0 0 0 0 0
8| x 2 1 2 x 1 0 0 1 1 1 0 0 0 0 0
9| 2 3 x 2 1 1 0 0 1 1 1 0 0 0 0 0
0| ? x 2 1 0 0 0 1 2 x 1 0 0 0 0 0
1| 2 2 2 0 0 0 0 1 x 2 2 1 1 0 0 0
2| 2 x 2 0 0 0 0 1 1 1 1 x 1 0 0 0
3| 2 x 2 0 0 0 0 0 0 0 2 2 2 0 0 0
4| 1 1 2 1 1 0 0 0 0 0 1 x 1 0 0 0
5| 0 0 1 x 1 0 0 0 0 0 1 1 1 0 0 0

-----  Uncover (10,0)  -----
Congratulation!!!!!!!
You Win
   0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
  --------------------------------
0| x 2 1 1 0 0 0 1 1 1 1 x 1 2 x 2
1| 1 2 x 2 1 1 0 1 x 1 1 1 1 2 x 2
2| 0 1 1 2 x 1 0 2 2 3 1 1 0 1 1 1
3| 0 0 0 1 1 1 0 1 x 3 x 3 1 0 0 0
4| 0 0 0 1 1 1 0 1 1 3 x x 1 0 0 0
5| 1 1 0 1 x 1 0 0 0 1 2 2 1 0 0 0
6| x 1 0 1 1 1 0 0 1 1 1 0 0 0 0 0
7| 2 2 0 1 1 1 0 0 1 x 1 0 0 0 0 0
8| x 2 1 2 x 1 0 0 1 1 1 0 0 0 0 0
9| 2 3 x 2 1 1 0 0 1 1 1 0 0 0 0 0
0| 1 x 2 1 0 0 0 1 2 x 1 0 0 0 0 0
1| 2 2 2 0 0 0 0 1 x 2 2 1 1 0 0 0
2| 2 x 2 0 0 0 0 1 1 1 1 x 1 0 0 0
3| 2 x 2 0 0 0 0 0 0 0 2 2 2 0 0 0
4| 1 1 2 1 1 0 0 0 0 0 1 x 1 0 0 0
```

level 3 (16x30 with 99 mines):



```
-----  Flag (15,28)  -----
   0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9
  -------------------------------------------------------------
0| 0 1 2 2 1 1 1 1 1 x 2 x 1 0 1 x 1 1 x 3 3 x 3 x 2 0 1 2 3 x
1| 0 1 x x 1 1 x 1 1 1 2 1 1 0 1 2 2 2 3 x x 2 3 x 3 1 2 x x 2
2| 0 2 3 4 2 3 2 2 0 0 1 1 1 0 0 2 x 3 3 x 3 1 1 1 2 x 2 2 2 1
3| 0 1 x 2 x 2 x 2 1 1 2 x 1 0 0 2 x 4 x 2 1 0 0 0 1 1 1 0 0 0
4| 1 2 1 2 1 2 2 x 2 3 x 4 2 1 0 2 3 x 2 1 0 0 1 1 1 0 0 0 0 0
5| x 1 0 0 0 1 2 3 x 3 x 3 x 1 0 1 x 2 2 2 2 1 2 x 2 0 0 0 1 1
6| 1 1 0 0 0 1 x 2 1 2 1 2 1 1 0 1 1 1 1 x x 3 3 x 3 1 0 0 1 x
7| 0 0 0 0 0 2 2 3 1 2 1 2 1 2 1 2 1 1 1 4 x 5 x 5 x 3 1 0 1 1
8| 0 0 1 1 1 1 x 2 x 2 x 2 x 2 x 2 x 3 2 3 x 5 x 4 x x 3 2 1 0
9| 1 2 3 x 3 3 2 2 2 3 3 3 2 2 1 2 2 x x 2 2 x 2 2 3 4 x x 1 0
0| 1 x x 3 x x 2 0 1 x 3 x 2 1 1 0 1 3 4 3 2 1 1 0 1 x 4 3 2 1
1| 1 3 3 4 4 x 2 1 2 4 x 4 3 x 1 1 1 3 x x 1 0 0 1 2 3 x 1 2 x
2| 1 3 x 3 x 2 1 2 x 4 x 4 x 3 1 1 x 3 x 4 2 1 0 1 x 2 1 1 2 x
3| x 3 x 4 2 1 1 3 x 3 1 3 x 2 1 2 3 4 3 3 x 1 1 2 2 2 1 2 2 2
4| 1 2 2 x 2 1 1 x 2 2 1 2 1 1 1 x 3 x x 2 1 2 2 x 1 1 x 4 x 2
5| 0 0 1 2 x 1 1 1 1 1 x 1 0 0 1 2 x 3 2 1 0 1 x 2 1 1 2 x x ?

-----  Uncover (15,29)  -----
Congratulation!!!!!!!
You Win
   0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9
  -------------------------------------------------------------
0| 0 1 2 2 1 1 1 1 1 x 2 x 1 0 1 x 1 1 x 3 3 x 3 x 2 0 1 2 3 x
1| 0 1 x x 1 1 x 1 1 1 2 1 1 0 1 2 2 2 3 x x 2 3 x 3 1 2 x x 2
2| 0 2 3 4 2 3 2 2 0 0 1 1 1 0 0 2 x 3 3 x 3 1 1 1 2 x 2 2 2 1
3| 0 1 x 2 x 2 x 2 1 1 2 x 1 0 0 2 x 4 x 2 1 0 0 0 1 1 1 0 0 0
4| 1 2 1 2 1 2 2 x 2 3 x 4 2 1 0 2 3 x 2 1 0 0 1 1 1 0 0 0 0 0
5| x 1 0 0 0 1 2 3 x 3 x 3 x 1 0 1 x 2 2 2 2 1 2 x 2 0 0 0 1 1
6| 1 1 0 0 0 1 x 2 1 2 1 2 1 1 0 1 1 1 1 x x 3 3 x 3 1 0 0 1 x
7| 0 0 0 0 0 2 2 3 1 2 1 2 1 2 1 2 1 1 1 4 x 5 x 5 x 3 1 0 1 1
8| 0 0 1 1 1 1 x 2 x 2 x 2 x 2 x 2 x 3 2 3 x 5 x 4 x x 3 2 1 0
9| 1 2 3 x 3 3 2 2 2 3 3 3 2 2 1 2 2 x x 2 2 x 2 2 3 4 x x 1 0
0| 1 x x 3 x x 2 0 1 x 3 x 2 1 1 0 1 3 4 3 2 1 1 0 1 x 4 3 2 1
1| 1 3 3 4 4 x 2 1 2 4 x 4 3 x 1 1 1 3 x x 1 0 0 1 2 3 x 1 2 x
2| 1 3 x 3 x 2 1 2 x 4 x 4 x 3 1 1 x 3 x 4 2 1 0 1 x 2 1 1 2 x
3| x 3 x 4 2 1 1 3 x 3 1 3 x 2 1 2 3 4 3 3 x 1 1 2 2 2 1 2 2 2
4| 1 2 2 x 2 1 1 x 2 2 1 2 1 1 1 x 3 x x 2 1 2 2 x 1 1 x 4 x 2
5| 0 0 1 2 x 1 1 1 1 1 x 1 0 0 1 2 x 3 2 1 0 1 x 2 1 1 2 x x 2
```

Well, to be honest, it is lucky in level 3, many times the program stuck because no more explicit information to do inference.
Like the case below.



```
-----  Flag (13,19)  -----
   0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9
  -------------------------------------------------------------
0| ? ? ? ? x 2 0 1 1 2 1 1 0 1 x x 1 2 x 2 0 0 1 x 1 0 1 1 1 0
1| ? ? x 4 x 2 0 1 x 3 x 2 0 1 2 3 2 3 x 3 2 1 2 2 2 1 1 x 2 1
2| ? ? 3 3 1 2 1 2 1 3 x 2 0 0 0 1 x 3 3 x 2 x 2 2 x 2 2 3 x 2
3| ? ? x 2 0 2 x 2 0 2 2 2 0 0 0 1 1 2 x 2 2 1 2 x 3 x 1 2 x 2
4| 1 3 x 2 0 3 x 4 1 2 x 2 1 1 0 0 0 1 1 1 0 0 1 2 3 3 2 2 1 1
5| 1 2 2 1 0 2 x 3 x 3 2 2 x 2 1 1 0 0 0 0 1 1 1 1 x 2 x 1 0 0
6| 2 x 3 1 1 1 1 3 4 x 3 2 1 2 x 1 0 0 0 0 1 x 1 1 1 2 1 1 1 1
7| 2 x 4 x 2 0 0 1 x x x 1 0 1 2 3 2 1 1 1 3 2 3 1 1 0 0 0 1 x
8| 1 1 3 x 3 1 0 1 2 3 3 3 2 2 2 x x 1 1 x 2 x 3 x 1 0 0 0 1 1
9| 0 1 2 4 x 2 1 1 1 0 1 x x 2 x 4 4 3 3 3 3 3 x 2 2 1 1 0 1 1
0| 1 2 x 3 x 3 2 x 1 0 1 2 2 2 2 x 2 x x 5 x 4 2 3 2 x 2 1 1 x
1| 2 x 2 2 2 x 3 3 2 2 1 1 0 0 2 2 3 3 x x x 4 x 2 x 3 x 1 1 1
2| x 2 1 0 1 3 x 3 x 2 x 1 1 2 4 x 3 2 4 ? x 3 1 3 2 3 2 2 1 0
3| 1 1 0 0 0 2 x 3 1 2 2 2 2 x x x ? ? 4 x 3 1 0 1 x 1 1 x 1 0
4| 0 1 1 1 0 1 2 2 1 0 1 x 3 4 ? ? ? ? ? ? 2 0 1 2 2 2 2 2 1 0
5| 0 1 ? 1 0 0 1 x 1 0 1 2 x 2 ? ? ? ? ? ? 1 0 1 x 1 1 x 1 0 0

-----  Flag (15,2)  -----
   0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9
  -------------------------------------------------------------
0| ? ? ? ? x 2 0 1 1 2 1 1 0 1 x x 1 2 x 2 0 0 1 x 1 0 1 1 1 0
1| ? ? x 4 x 2 0 1 x 3 x 2 0 1 2 3 2 3 x 3 2 1 2 2 2 1 1 x 2 1
2| ? ? 3 3 1 2 1 2 1 3 x 2 0 0 0 1 x 3 3 x 2 x 2 2 x 2 2 3 x 2
3| ? ? x 2 0 2 x 2 0 2 2 2 0 0 0 1 1 2 x 2 2 1 2 x 3 x 1 2 x 2
4| 1 3 x 2 0 3 x 4 1 2 x 2 1 1 0 0 0 1 1 1 0 0 1 2 3 3 2 2 1 1
5| 1 2 2 1 0 2 x 3 x 3 2 2 x 2 1 1 0 0 0 0 1 1 1 1 x 2 x 1 0 0
6| 2 x 3 1 1 1 1 3 4 x 3 2 1 2 x 1 0 0 0 0 1 x 1 1 1 2 1 1 1 1
7| 2 x 4 x 2 0 0 1 x x x 1 0 1 2 3 2 1 1 1 3 2 3 1 1 0 0 0 1 x
8| 1 1 3 x 3 1 0 1 2 3 3 3 2 2 2 x x 1 1 x 2 x 3 x 1 0 0 0 1 1
9| 0 1 2 4 x 2 1 1 1 0 1 x x 2 x 4 4 3 3 3 3 3 x 2 2 1 1 0 1 1
0| 1 2 x 3 x 3 2 x 1 0 1 2 2 2 2 x 2 x x 5 x 4 2 3 2 x 2 1 1 x
1| 2 x 2 2 2 x 3 3 2 2 1 1 0 0 2 2 3 3 x x x 4 x 2 x 3 x 1 1 1
2| x 2 1 0 1 3 x 3 x 2 x 1 1 2 4 x 3 2 4 ? x 3 1 3 2 3 2 2 1 0
3| 1 1 0 0 0 2 x 3 1 2 2 2 2 x x x ? ? 4 x 3 1 0 1 x 1 1 x 1 0
4| 0 1 1 1 0 1 2 2 1 0 1 x 3 4 ? ? ? ? ? ? 2 0 1 2 2 2 2 2 1 0
5| 0 1 x 1 0 0 1 x 1 0 1 2 x 2 ? ? ? ? ? ? 1 0 1 x 1 1 x 1 0 0
```

Total steps:

	1	2	3	4	5	6	7	8	avg
Lv1	44	31	34	29	34	27	24	30	31.6
Lv2	72	56	65	-	55	70	58	61	62.4
Lv3	333	357	-	-	-	-	-	-	-

- symbol means stuck occurs.
Lv3 doesn't give a avg. step because the game can reach goal about every 10 times of tests.

Observation of test:

To write this program, I play Minesweeper on the internet to determine the way I implement this program. The total steps of the program are almost equal to the steps I play. Hence, I think the rules in this program can make this program as an amateur player.
And it is also reasonable that so many stuck in Lv3, when I play game on the internet, I need to guess a position because no more information for me to claim a grid is safe or not. In a round of game, about 4-5 times of guess is necessary. This program doesn't have the function of guess, so it will be stuck at the time.

Stuck and Guessing:

To my observation, when stuck occurs, the program will keep doing matching but the size of KB won't modify before and after matching. If the program keeps doing matching and can't get a single-literal clause for many iterations, we can call it stuck.
I don't deal with this situation in my program. I think if stuck occurs, the KB is useless, we can flush the KB and guess a position and claim it is safe to get more information from Game Master, although the guess may cause the game terminates because the grid we guess has mine inside.

And the way of guessing, I think I will directly find the most possible safe grid to guess, which means 4 grids with 1 mine (rate of death = 0.25) has higher priority than 3 grids with 1 mine (rate of death = 0.33). And there is one thing should be notice, the ratio of mines to grids is 0.20 in Lv3 which means the rate of death = 0.20 if we do the random guess. If we calculate rate of death from KB0 are all larger than 0.20 in Lv3, I think I will do the random guess and leave the result to God though I'm not a religious person.

Discussion: Use method in Assignment 2:

Method in assignment 2 (Consistency check) may be better than using propositional logic.
Consider a simple case.

1	1	1
x	y	z

Propositional logic:
1. $(x \lor y), (\neg x \lor \neg y), (\neg y \lor \neg z), (\neg x \lor \neg z), (y \lor z)$
then do 20 times pairwise matching to obtain
2. $(x \lor y), (\neg x \lor \neg y), (\neg y \lor \neg z), (\neg x \lor \neg z), (y \lor z), (x \lor \neg z), (y \lor \neg z), (\neg x \lor z)$
then do 56 times pairwise matching to obtain
3. $(\neg z)$, … => $z = 0$

Consistency check:
1. $x+y = 1$, $x+y+z = 1$, $y+z=1$
y's degree=3, y assign first
2. y assigns 0, check 3 constrains, get contradiction, do backtracking.
3. y assigns 1, check 3 constrains, all constrains solve.

In propositional logic, matching is necessary when no single-literal clause exist, however it is very waste time.
Generally, every hints in propositional logic generate more than one constrain but every hints generate only single constrain in consistency check.

Appendix:

main.cpp:

```
#include <iostream>
#include <cstdlib> // random
#include <ctime> // time
#include <vector>
#include <algorithm>
#include <cmath>
#include <queue>
#include <set>

using namespace std;
class Minesweeper{
private:
    vector<vector<int> > gField; // GM field
    int unsolved;
public:
    vector<vector<int> > pField; // player field, KB0
    int n,m,NumofMines;
    Minesweeper(int level);
    void PrintField(bool auth); // if auth is true print gm field
    int GameStart(); // return a safe start grid x * 100 + y
    int Uncover(int x,int y); // player uncover the covered grid (x,y), GM
modify player field
    void Manual(); // Manual module, for DEBUG or for fun
};
int dx[] = {-1,-1,-1,0,0,1,1,1},dy[] = {-1,0,1,-1,1,-1,0,1}; // direction
void Match(set<vector<int> > & KB);
void Clause_Generate(int x,int y,Minesweeper & Game,set<vector<int>
> & KB);
bool Claim(int pos,Minesweeper & Game,set<vector<int> > &
KB,vector<vector<bool> > & inKB);
int main(int argc, char const *argv[]){
    int lv;
    printf("Level selection 1~3 : ");
    scanf("%d",&lv);
    Minesweeper Game(lv);
    int pos = Game.GameStart(); // GM will tell player a safe initial place
    if(pos == -1){ // no safe position
        printf("Exception occurs, contact GM.\n");
        Game.PrintField(true);
        return 0;
    }
    printf("-----     Game start     -----\n");
    Game.PrintField(false);

    set<vector<int> > KB;
    vector<vector<bool> > inKB(Game.n,vector<bool>(Game.m,false));
    KB.insert(vector<int>(1,pos));
    while(!KB.empty()){
        set<vector<int> >::iterator it;
        for(it = KB.begin();it != KB.end();it++)
```

```
            if((*it).size() == 1){ // single literal clause exist
                if(!Claim((*it)[0],Game,KB,inKB))
                    return 0; // game terminsted by GM
                break;
            }
        if(it == KB.end()) // single literal clause not exist
            Match(KB);
    }
    return 0;
}
Minesweeper::Minesweeper(int level){
    if(level == 1){
        gField = vector<vector<int> >(9,vector<int>(9));
        n = m = 9;
        NumofMines = 10;
    }
    else if(level == 2){
        gField = vector<vector<int> >(16,vector<int>(16));
        n = m = 16;
        NumofMines = 25;
    }
    else{
        gField = vector<vector<int> >(16,vector<int>(30));
        n = 16;
        m = 30;
        NumofMines = 99;
    }
    pField = gField;
    unsolved = n * m - NumofMines; // the remain blocks need to be
uncovered
    // set mines
    srand(time(NULL));
    int remain_mine = NumofMines;
    while(remain_mine){
        int x = rand() % n,y = rand() % m;
        if(gField[x][y] == -1)
            continue;
        gField[x][y] = -1;
```

```
            for(int di = 0;di < 8;di++){
                  int nx = x + dx[di],ny = y + dy[di];
                  if(0 <= nx && nx < n && 0 <= ny && ny < m &&
gField[nx][ny] != -1)
                        gField[nx][ny]++;
            }
            remain_mine--;
      }
}
void Minesweeper::PrintField(bool auth){
      vector<vector<int> > & Field = auth?gField:pField;
      printf("      ");
      for(int j = 0;j < m;j++)
            printf("%d ",j % 10);
      printf("\n    -");
      for(int j = 0;j < m;j++)
            printf("--");
      printf("\n");
      for(int i = 0;i < n;i++){
            printf("%d| ",i % 10);
            for(int j = 0;j < m;j++){
                  if(Field[i][j] == -2)
                        printf("? ");
                  else if(Field[i][j] == -1)
                        printf("x ");
                  else
                        printf("%d ",Field[i][j]);
            }
            printf("\n");
      }
      printf("\n");
}
int Minesweeper::GameStart(){
      int position = -1;
      for(int i = 0;i < n;i++)
            for(int j = 0;j < m;j++){
                  pField[i][j] = -2; // covered
                  if(gField[i][j] == 0 && position == -1)
```

```
                    position = i * 100 + j;
            }
        return position; // claim it is safe
}
int Minesweeper::Uncover(int x,int y){
        // return 0: game is solved or player lose, terminates the game
        //              1: game should be continued
        printf("-----    Uncover (%d,%d)    -----\n",x,y);
        if(gField[x][y] == -1){ // fail case, shouldn't happen
            printf("Bang!!!!!!!\nYou Lose\n");
            PrintField(true);
            return 0;
        }
        else{
            queue<int> q;
            q.push(x * 100 + y);
            while(!q.empty()){
                x = q.front() / 100;
                y = q.front() % 100;
                q.pop();
                if(pField[x][y] != -2)
                    continue;
                pField[x][y] = gField[x][y];
                unsolved--;
                if(!gField[x][y])
                    for(int di = 0;di < 8;di++){
                        int nx = x + dx[di],ny = y + dy[di];
                        if(0 <= nx && nx < n && 0 <= ny && ny < m &&
gField[nx][ny] != -1 && pField[nx][ny] == -2)
                            q.push(nx * 100 + ny);
                    }
            }
        }
        // goal test
        if(!unsolved){ // game clear, player wins
            printf("Congratulation!!!!!!!\nYou Win\n");
            PrintField(true);
            printf("Total Step :%d\n",step);
            return 0;
        }
        PrintField(false);
        return 1;
}
void Minesweeper::Manual(){
        int x,y;
        do{
            printf("Input the grid should be uncovered: ");
            scanf("%d%d",&x,&y);
            printf("Uncover (%d,%d)\n",x,y);
            if(!Uncover(x,y))
                break;
            PrintField(false);
        }while(true);
}
void Clause_Generate(int x,int y,Minesweeper & Game,set<vector<int>
> & KB){
        if(Game.pField[x][y] == 0 || Game.pField[x][y] == -1)
            return;
        int N = Game.pField[x][y];
        vector<int> Var;
        for(int di = 0;di < 8;di++){
            int nx = x + dx[di],ny = y + dy[di];
            if(0 <= nx && nx < Game.n && 0 <= ny && ny < Game.m &&
Game.pField[nx][ny] == -2)
                Var.push_back(nx * 100 + ny);
            if(0 <= nx && nx < Game.n && 0 <= ny && ny < Game.m &&
Game.pField[nx][ny] == -1)
                N--;
        }
        if(N == Var.size()){ // all variable = -1
            for(int vi = 0;vi < Var.size();vi++)
                KB.insert(vector<int>(1,10000 + Var[vi]));
            return;
        }
        if(N == 0){ // all variable = 0
            for(int vi = 0;vi < Var.size();vi++)
```

```
            KB.insert(vector<int>(1,Var[vi]));
        return;
    }
    vector<int> clause(2,-1);
    if(Var.size() - N == 1){ // pos clause
        for(int vi = 0;vi < Var.size();vi++){
            clause[0] = Var[vi] + 10000;
            for(int vj = vi + 1;vj < Var.size();vj++){
                clause[1] = Var[vj] + 10000;
                sort(clause.begin(),clause.end());
                KB.insert(clause);
            }
        }
    }
    if(N == 1){ // neg clause
        for(int vi = 0;vi < Var.size();vi++){
            clause[0] = Var[vi];
            for(int vj = vi + 1;vj < Var.size();vj++){
                clause[1] = Var[vj];
                sort(clause.begin(),clause.end());
                KB.insert(clause);
            }
        }
    }
}
bool Claim(int pos,Minesweeper & Game,set<vector<int> > &
KB,vector<vector<bool> > & inKB){
    bool isMine = false;
    if(pos / 10000){ //
        pos %= 10000;
        isMine = true;
    }
    int x = pos / 100,y = pos % 100;
    if(isMine){
        printf("-----    Flag (%d,%d)    -----\n",x,y);
        Game.pField[x][y] = -1;
        Game.PrintField(false);
    }
```

```
	else{
		bool rt = Game.Uncover(x,y);
		if(!rt)
			return rt;
	}

	// matching KB with KB0
	set<vector<int> > nKB;
	for(auto nit = KB.begin();nit != KB.end();nit++){
		vector<int> clause;
		for(int vi = 0;vi < (*nit).size();vi++){
			bool M = false,Mp;
			int lit = (*nit)[vi];
			if(lit / 10000){
				M = true;
				lit %= 10000;
			}
			if(Game.pField[lit / 100][lit % 100] == -2){
				clause.push_back(lit + (M?10000:0));
				continue;
			}
			Mp = Game.pField[lit / 100][lit % 100] == -1;
			if(M == Mp){ // always true clause -> drop
				clause.clear();
				break;
			}
		}
		if(!clause.empty()){
			sort(clause.begin(),clause.end());
			nKB.insert(clause);
		}
	}
	KB = nKB; // replace

	// generating new clause
	queue<int> q;
	vector<vector<bool> >
inQueue(Game.n,vector<bool>(Game.m,false));
```

```
    q.push(x * 100 + y);
    inQueue[x][y] = true;
    while(!q.empty()){
        x = q.front() / 100;
        y = q.front() % 100;
        q.pop();
        Clause_Generate(x,y,Game,KB);

        for(int di = 0;di < 8;di++){
            int nx = x + dx[di],ny = y + dy[di];
            if(0 <= nx && nx < Game.n && 0 <= ny && ny < Game.m
&& !inQueue[nx][ny] && Game.pField[nx][ny] != -2){
                q.push(nx * 100 + ny);
                inQueue[nx][ny] = true;
            }
        }
    }
    return true;
}
void Match(set<vector<int> > & KB){
    set<vector<int> > nKB;

    vector<vector<int> > vKB;
    for(auto it = KB.begin();it != KB.end();it++)
        vKB.push_back(*it);
    for(int i = 0;i < vKB.size();i++){
        for(int j = 0;j < vKB.size();j++){
            vector<int> clause;
            for(int vi = 0;vi < vKB[i].size();vi++){
                bool pass = true;
                for(int vj = 0;vj < vKB[j].size();vj++)
                    if(abs(vKB[i][vi] - vKB[j][vj]) == 10000)
                        pass = false;
                if(pass)
                    clause.push_back(vKB[i][vi]);
            }
            for(int vj = 0;vj < vKB[j].size();vj++){
                bool pass = true;
```

```
                    for(int vi = 0;vi < vKB[i].size();vi++)
                         if(abs(vKB[i][vi] - vKB[j][vj]) == 10000 ||
vKB[i][vi] - vKB[j][vj] == 0)
                              pass = false;
                    if(pass)
                         clause.push_back(vKB[j][vj]);
               }
               if(clause.size() == 1 || clause.size() == 2){
                    sort(clause.begin(),clause.end());
                    nKB.insert(clause);
               }
          }
     }
     KB = nKB; // replace
}
```