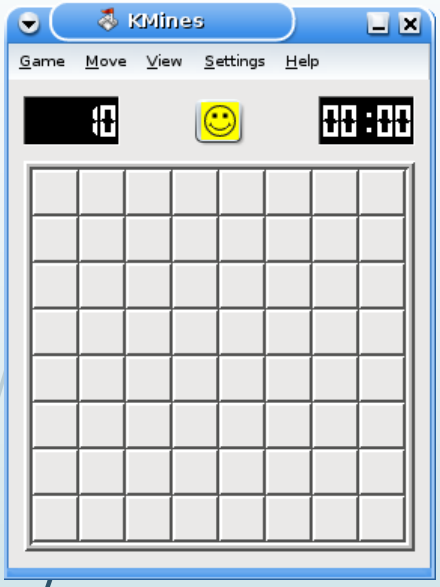
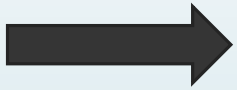
Design a Minesweeper

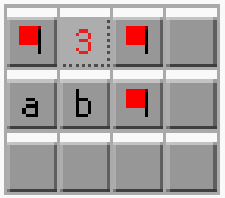
Speaker : Cheng-Zse Wu
Advisor :Jen-Wei Huang

The Rules of Minesweeper



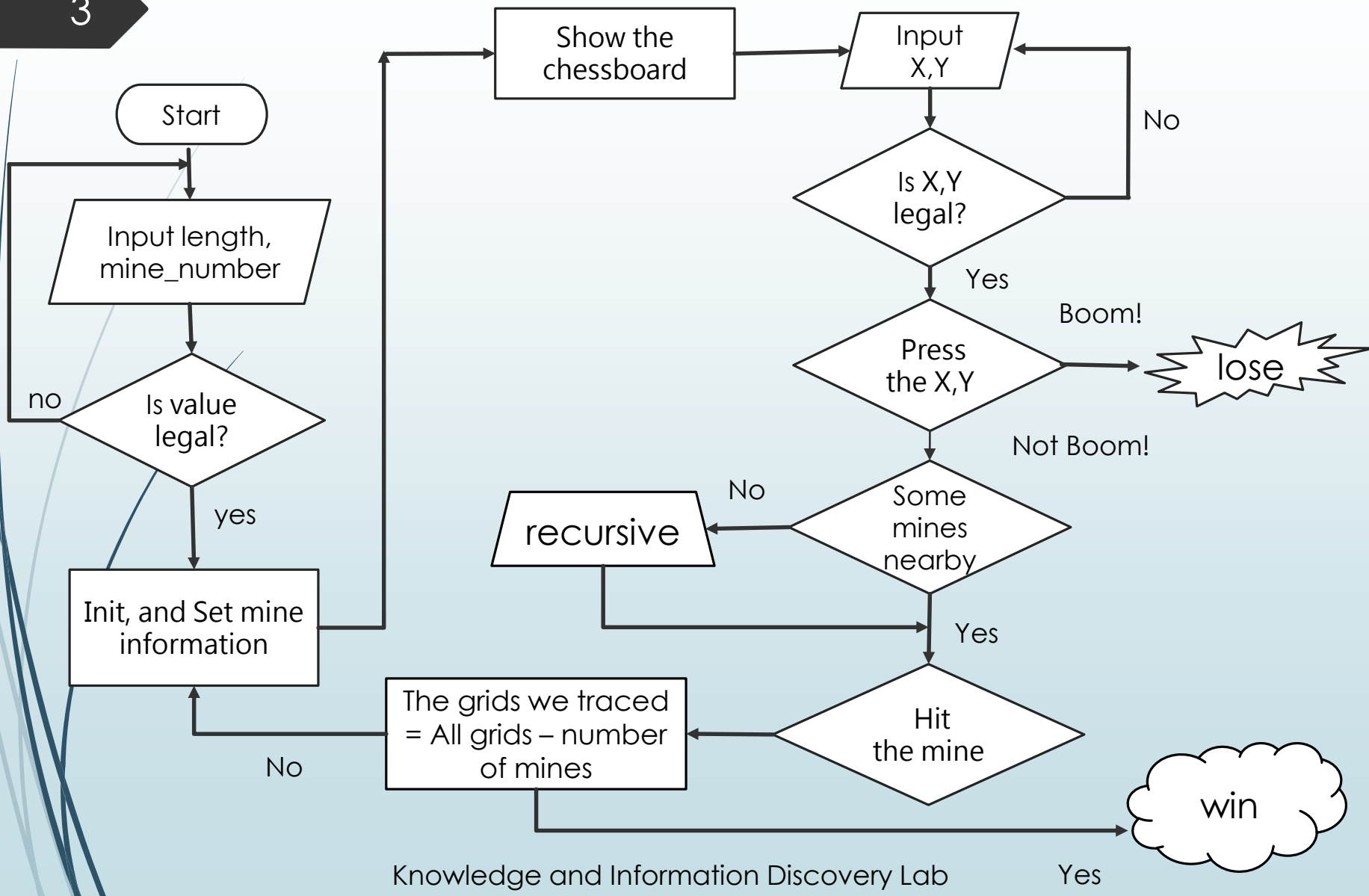


範例1

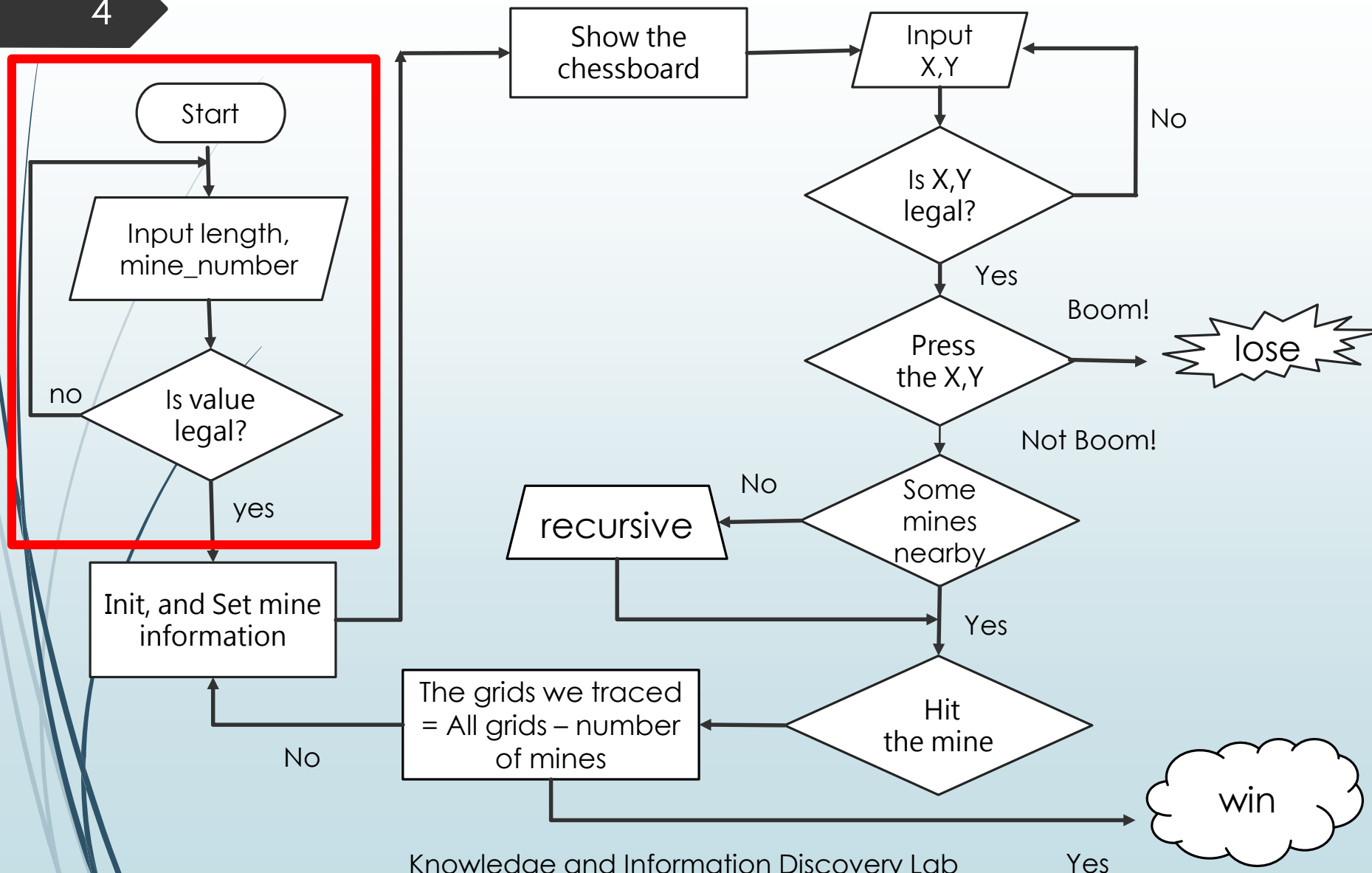


a和b沒有地雷，可輕易點開，因為已有3顆地雷滿足數字3

Flow chart


Start
Input length, mine_number
Is value legal?
no
yes
Init, and Set mine information
Show the chessboard
Input X,Y
Is X,Y legal?
No
Yes
Press the X,Y
Boom!
lose
Not Boom!
Some mines nearby
No
recursive
Yes
Hit the mine
The grids we traced = All grids – number of mines
No
win
Yes

Flow chart


Start
Input length, mine_number
Is value legal?
no
yes
Init, and Set mine information
Show the chessboard
Input X,Y
Is X,Y legal?
No
Yes
Press the X,Y
Boom!
lose
Not Boom!
Some mines nearby
No
recursive
Yes
Hit the mine
The grids we traced = All grids – number of mines
No
win
Yes

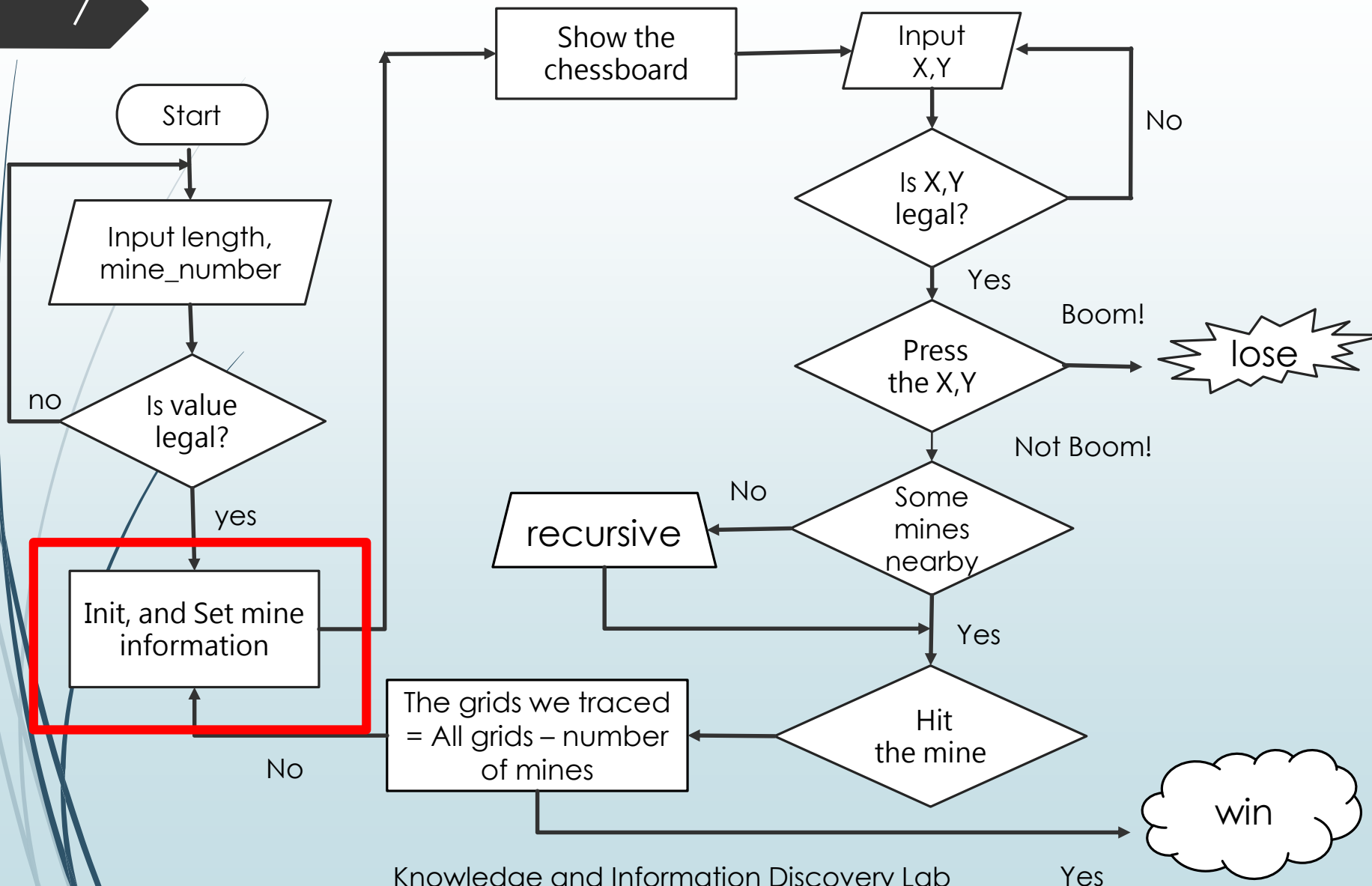
Start() function


```
232 void Start(){           //start game
233     do{
234         input_error = 0;
235         cout << "Enter the board length  :";
236         cin >> boardlength;
237         cout << "Enter the mine number:" << "(" << "0~" << boardlength*boardlength << ")  :";
238         cin >> mine_number;
239
240         /*11111111111111111111111111111111111test1111111111111111111111111111111111111111
241         please fill the blank.
242         When user's input value is illegal, then the process will ask user to retry.
243         Hint: using the if() and the argument "input_error" and use system("pause") to catch the screen.
244         1111111111111111111111111111111111111111111111111111111111111111111111111111111111*/
245
246
247         system("cls");//clean screen
248     } while (input_error);
249     Init();
250     bool coutinue = 1;
251     while (coutinue){
252         ShowMineboard(2);
253         Press();
254         coutinue = Check();
255     }
256     return;
257 }
```

Start() function


```
242 void Start(){           //start game
243     do{
244         input_error = 0;
245         cout << "Enter the board length(1~17)  :";
246         cin >> boardlength;
247         cout << "Enter the mine number:" << "(" << "0~" << boardlength*boardlength << ")  :";
248         cin >> mine_number;
249         if (boardlength<1 || mine_number<0 || mine_number>boardlength*boardlength)
250         {
251             cout << "input error, please try again!" << "\n";
252             input_error = 1;
253             system("pause");
254         }
255         system("cls");//clean screen
256     } while (input_error);
257
258     Init();
259     bool contiune = 1;
260     while (contiune){
261         ShowMineboard(2);
262         Press();
263         contiune = Check();
264     }
265     return;
266 }
```

Flow chart


Start
Input length, mine_number
Is value legal?
no
yes
Init, and Set mine information
Show the chessboard
Input X,Y
Is X,Y legal?
No
Yes
Press the X,Y
Boom!
lose
Not Boom!
Some mines nearby
No
recursive
Yes
Hit the mine
The grids we traced = All grids – number of mines
No
win
Yes

mine_info[x][y]

-2: wall

-1: mine

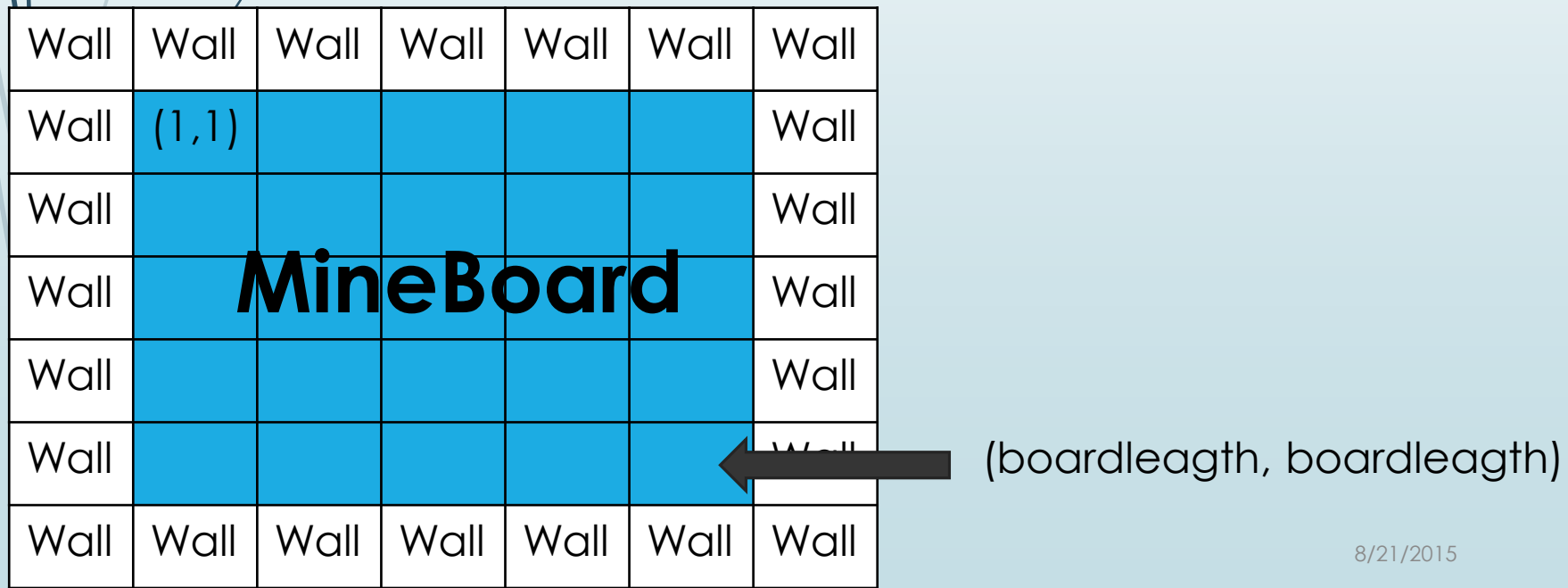
0: there is no mine around,

>0: how many mines around **with (x,y)**

map_press[x][y]

0: unstepped,

1: stepped



Init() function


```
void Init(){                                    //initialize
    int counter = 0;
    int i, j;
    for (i = 1; i <= boardlength; i++){   //initialize the mineboard
        for (j = 1; j <= boardlength; j++)
        {
            mine_info[i][j] = 0;
            map_press[i][j] = 0;
        }
    }
    for (i = 0; i <= boardlength + 1; i++){   //initialize the wall
        mine_info[i][0] = -2;
        mine_info[i][boardlength + 1] = -2;
    }
    for (j = 0; j <= boardlength + 1; j++){   //initialize the wall
        mine_info[0][j] = -2;
        mine_info[boardlength + 1][j] = -2;
    }
```

Init() function

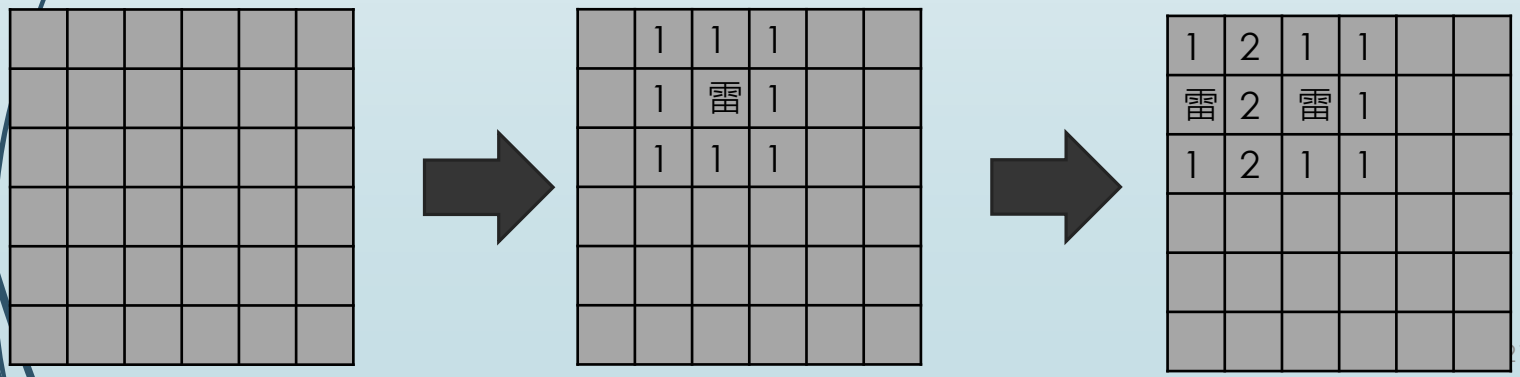

```
    srand(time(NULL));
    //22222222222222222222222test22222222222222222222222222222222222
    /*Please compelete the following code.
    Set n mines randomly on location (x, y), n = mine_number.
    Let x and y belong to 1 ~ boardlength.
    Hit: using the argument "counter" and %.
    The mine_info[][] can express the mine.
    */
    /*
    while (){  //control the mumber of mines
        int x = rand();
        int y = rand();
        if (){

            SetMineLable(x, y);  //do not change this line!!

        }
    }
    */
    //22222222222222222222222test22222222222222222222222222222222222
}
```

Init() function


```
179     srand(time(NULL));
180     while (counter < mine_number){          //set the mines on the board
181         int x = (rand() % boardlength) + 1;
182         int y = (rand() % boardlength) + 1;
183         if (mine_info[x][y] != -1){
184             mine_info[x][y] = -1;
185             SetMineLable(x, y);
186             counter++;
187         }
188     }
189 }
```

SetMineLable() function

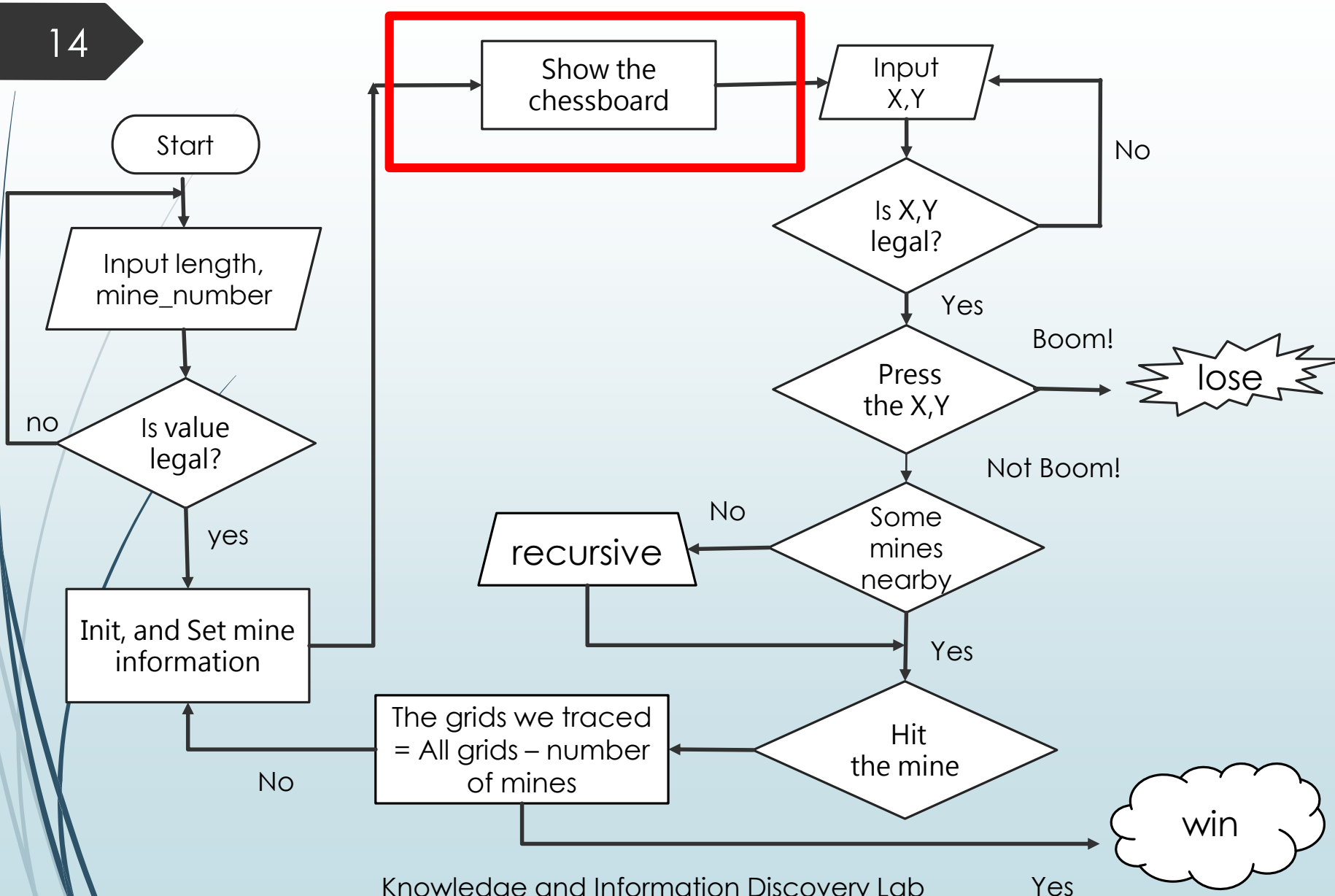

```
void SetMineLable(int x, int y) //update how many mines around
{
    //33333333333333333333test33333333333333333333
    /* check 8 location that around the (x, y),
    if the location is not mine and not wall then update the mine number
    Hit: using the mine_info[][]
    */


    //33333333333333333333test33333333333333333333
}
```



SetMineLable() function


```
72  void SetMineLable(int x, int y) //update how many mines around
73  {
74      if (mine_info[x - 1][y - 1] != -1 && mine_info[x - 1][y - 1] != -2){ //if (x - 1, y - 1) is not a mine and not a wall
75          mine_info[x - 1][y - 1]++;//left-up mine number++
76      }
77      if (mine_info[x - 1][y] != -1 && mine_info[x - 1][y] != -2){
78          mine_info[x - 1][y]++;
79      }
80      if (mine_info[x - 1][y + 1] != -1 && mine_info[x - 1][y + 1] != -2){
81          mine_info[x - 1][y + 1]++;
82      }
83      if (mine_info[x][y - 1] != -1 && mine_info[x][y - 1] != -2){
84          mine_info[x][y - 1]++;
85      }
86      if (mine_info[x][y] != -1 && mine_info[x][y] != -2){
87          mine_info[x][y]++;
88      }
89      if (mine_info[x][y + 1] != -1 && mine_info[x][y + 1] != -2){
90          mine_info[x][y + 1]++;
91      }
92      if (mine_info[x + 1][y - 1] != -1 && mine_info[x + 1][y - 1] != -2){
93          mine_info[x + 1][y - 1]++;
94      }
95      if (mine_info[x + 1][y] != -1 && mine_info[x + 1][y] != -2){
96          mine_info[x + 1][y]++;
97      }
98      if (mine_info[x + 1][y + 1] != -1 && mine_info[x + 1][y + 1] != -2){
99          mine_info[x + 1][y + 1]++;
100     }
101 }
```

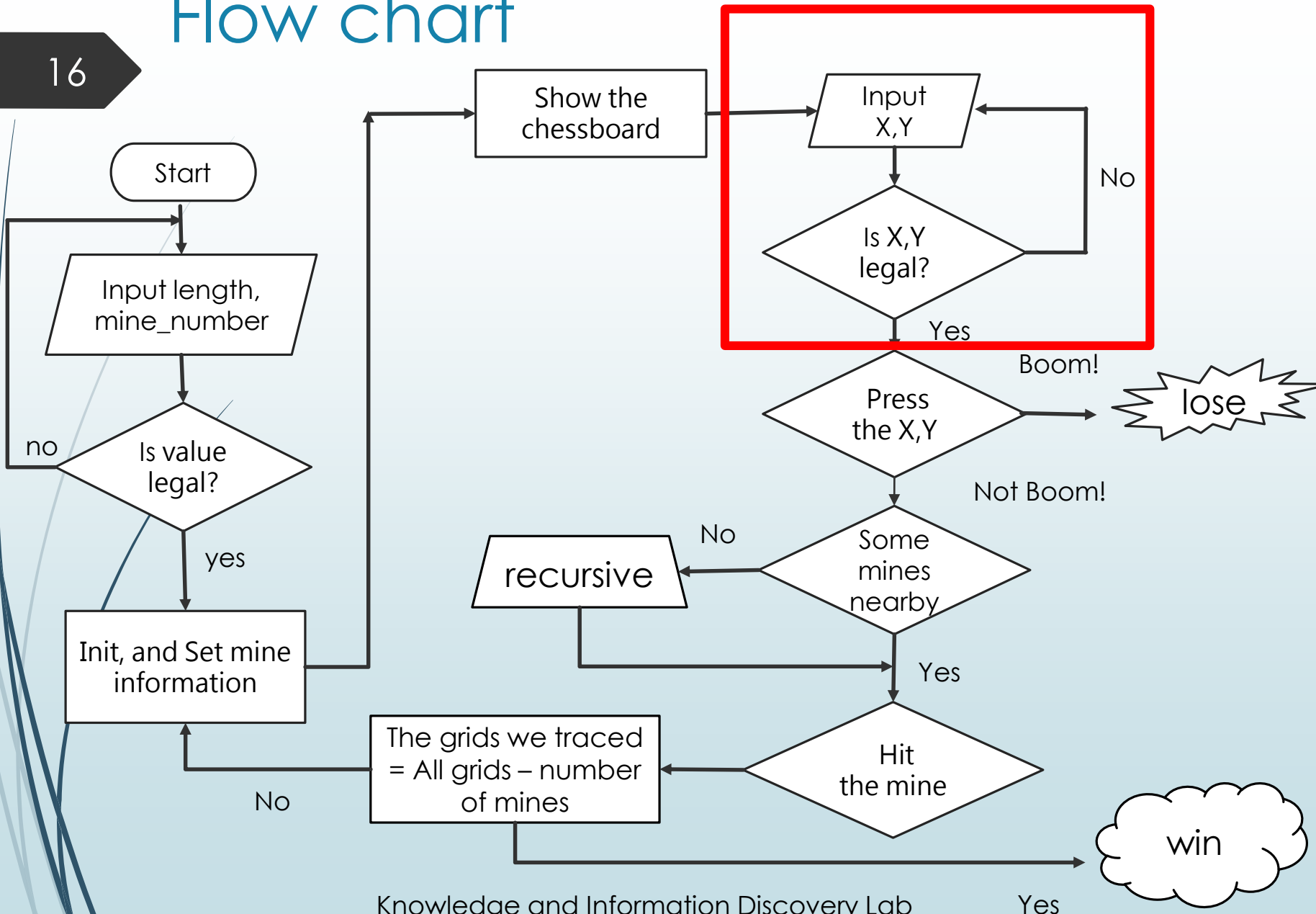
Flow chart


Start
Input length, mine_number
Is value legal?
no
yes
Init, and Set mine information
Show the chessboard
Input X,Y
Is X,Y legal?
No
Yes
Press the X,Y
Boom!
lose
Not Boom!
Some mines nearby
No
recursive
Yes
Hit the mine
The grids we traced = All grids – number of mines
No
Yes
win

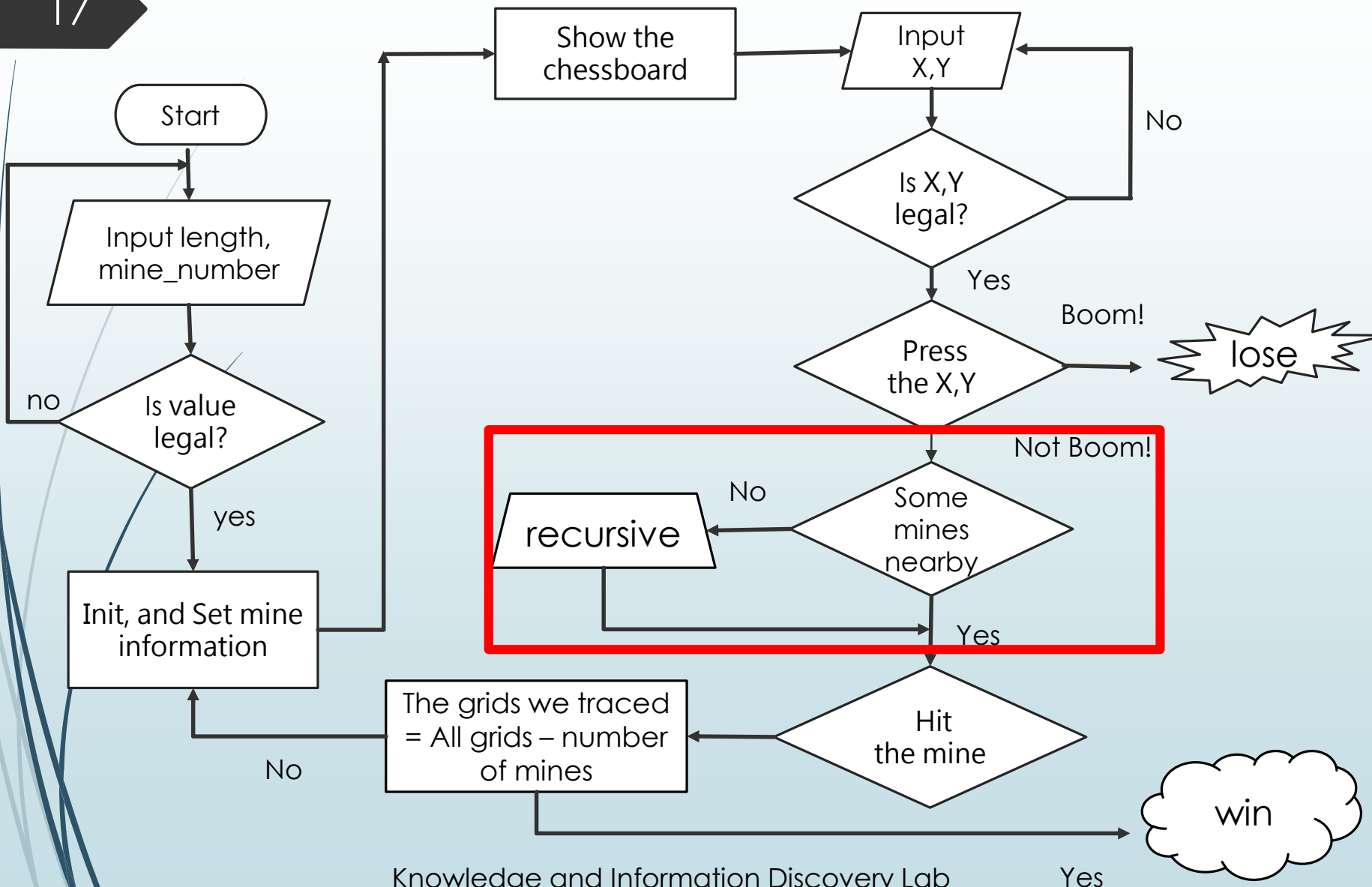
ShowMap() function

```
void ShowMineboard(int win){   //show the mineboard

    cout << "(y) ";
    for (int i = 0; i < boardlength; i++){
        cout << "__";
    }
    cout << "\n";
    for (int j = boardlength; j > 0; j--){
        if (j > 9)cout << j << " |";
        else cout << " " << j << " |";
        for (int i = 1; i <= boardlength; i++){
            if (map_press[i][j]){
                if (mine_info[i][j] == -1 && win != 1){
                    cout << "爆";
                }
                else if (mine_info[i][j] == -1 && win == 1){
                    cout << "雷";
                }
                else if (mine_info[i][j] == 0)  cout << "  ";
                else  cout << " " << mine_info[i][j];
            }
            else
                cout << "■";
        }
        cout << "| " << '\n';
    }
```

Flow chart

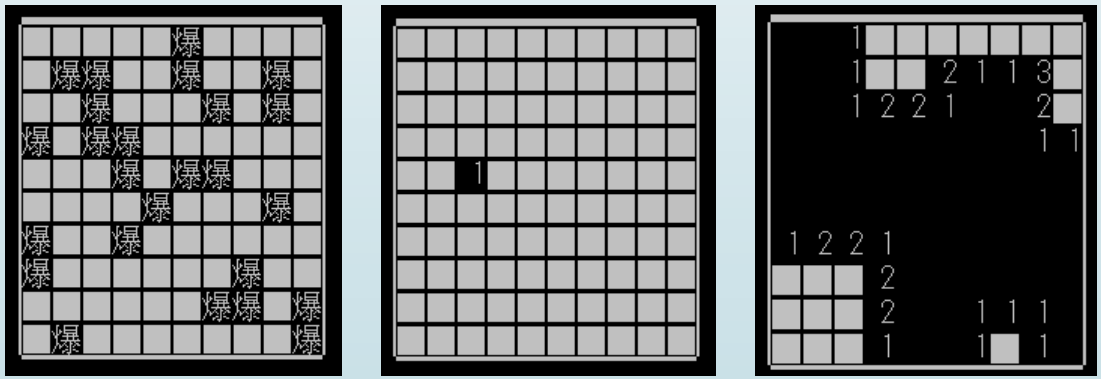

Start
Input length, mine_number
Is value legal?
no
yes
Init, and Set mine information
Show the chessboard
Input X,Y
Is X,Y legal?
No
Yes
Press the X,Y
Boom!
lose
Not Boom!
Some mines nearby
No
recursive
Yes
Hit the mine
The grids we traced = All grids – number of mines
No
Yes
win

Flow chart


Start
Input length, mine_number
Is value legal?
no
yes
Init, and Set mine information
Show the chessboard
Input X,Y
Is X,Y legal?
No
Yes
Press the X,Y
Boom!
lose
Not Boom!
Some mines nearby
No
recursive
Yes
Hit the mine
The grids we traced = All grids – number of mines
No
win
Yes

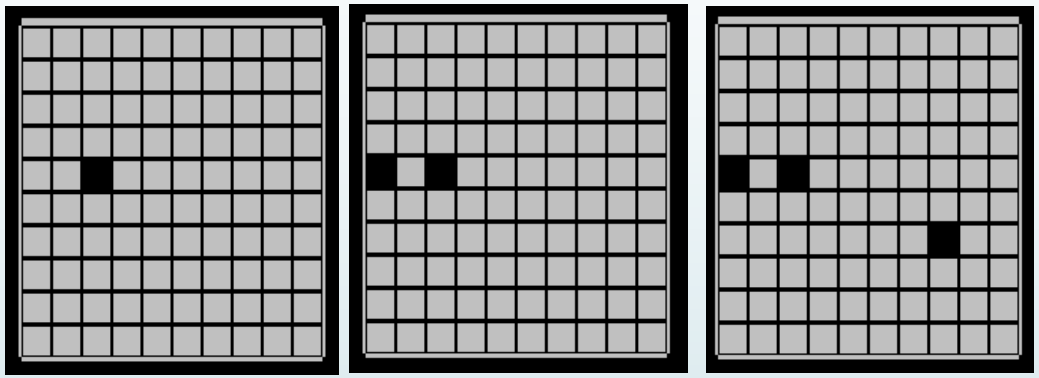
When the button is pressed

- We have to consider 3 cases.
  - Boom!
  - No explosion, but there are some bombs next the position.
  - No explosion, and no bomb nearby.

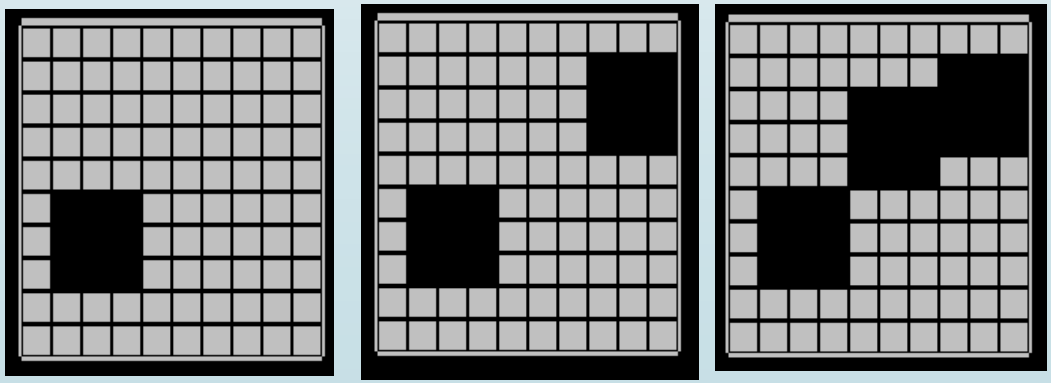


No bomb nearby

- Only press the button

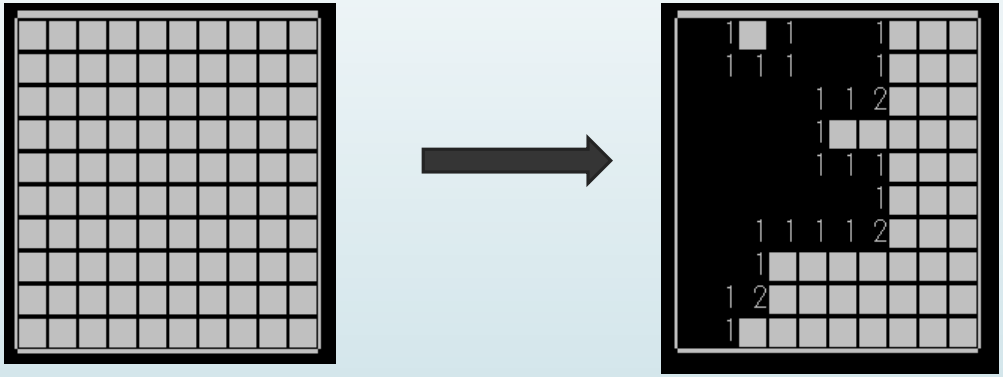


- Press other 8 buttons



No bomb nearby

- We need some information to continue the game, or it just guessing.



How can we do this just press the button once?

Press() function

```
void Press(){    //step
    input_error = 0;
    cout << "input x:  ";
    cin >> x;
    cout << "input y:  ";
    cin >> y;
    system("cls");
    if (0 < x && x <= boardlength && 0 < y && y <= boardlength)
        map_press[x][y] = 1;//step (x, y)
    else{
        input_error = 1;
        cout << "input error, please try again!" << "\n";
        system("pause");
        system("cls");
    }
    if (input_error == 0 && mine_info[x][y] == 0) NoMineAround(x, y);
    //if there is no mine on (x, y)，call NoMineAround()
}
```

No bomb nearby

- We have to use “Recursion”.

```
81  void NoMineAround(int x, int y){
82      /*444444444444444444444444444444444test444444444444444444444444444444444444
83      if the location was not pressed and it is not the wall then first we can set the location to be pressed
84      And we check the mine_info if it is zero then we run the NoMineAround again (Recursive)
85
86      if (){                                                  //if (x - 1, y - 1) unstepped and is not a wall
87                                                               //step (x - 1, y - 1)
88      if (){                                                  // no mine around (x - 1, y - 1)
89
90
91      444444444444444444444444444444444444test444444444444444444444444444444444444*/
92  }
```

No bomb nearby

- We have to use “Recursion”.

1	2	1	1		
雷	2	雷	1		
1	2	1	1	1	1
1	1		★	1	雷
雷	2	1	2	3	3
1	2	雷	2	雷	雷

1	2	1	1		
雷	2	雷	1		
1	2	1	1	1	1
1	1	★		1	雷
雷	2	1	2	3	3
1	2	雷	2	雷	雷

1	2	1	1		
雷	2	雷	1		
1	2	1	1	1	1
1	1		★	1	雷
雷	2	1	2	3	3
1	2	雷	2	雷	雷

1	2	1	1		
雷	2	雷	1		
1	2	1	1	1	1
1	1			1	雷
雷	2	1	2	3	3
1	2	雷	2	雷	雷

1	2	1	1		
雷	2	雷	1		
1	2	1	1	1	1
1	1		★	1	雷
雷	2	1	2	3	3
1	2	雷	2	雷	雷

1	2	1	1		
雷	2	雷	1		
1	2	1	1	1	1
1	1	★		1	雷
雷	2	1	2	3	3
1	2	雷	2	雷	雷

1	2	1	1		
雷	2	雷	1		
1	2	1	1	1	1
1	1		★	1	雷
雷	2	1	2	3	3
1	2	雷	2	雷	雷

1	2	1	1		
雷	2	雷	1		
1	2	1	1	1	1
1	1		★	1	雷
雷	2	1	2	3	3
1	2	雷	2	雷	雷

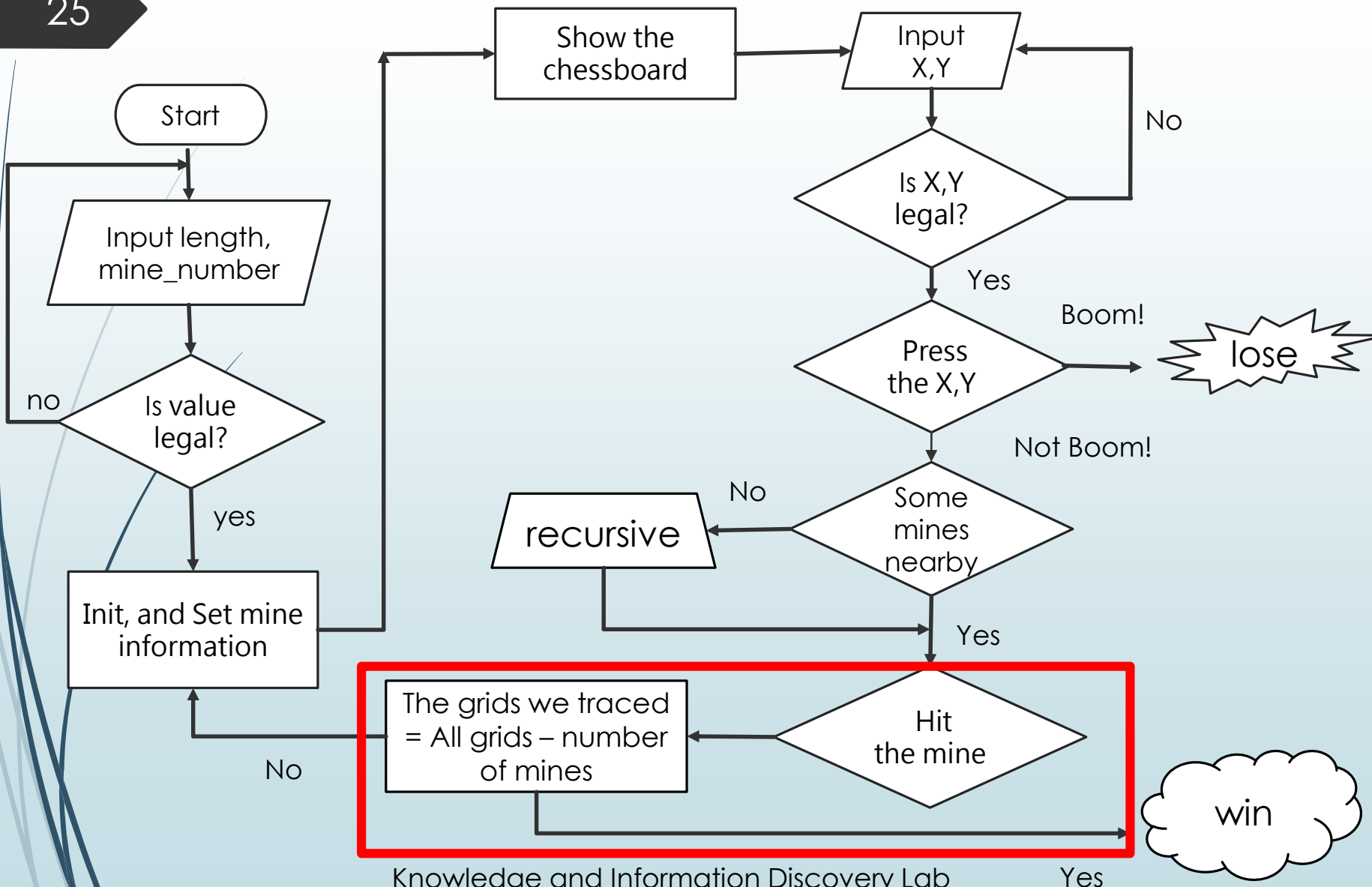
★=(x, y)的位置

No bomb nearby

- We have to use “Recursion”.

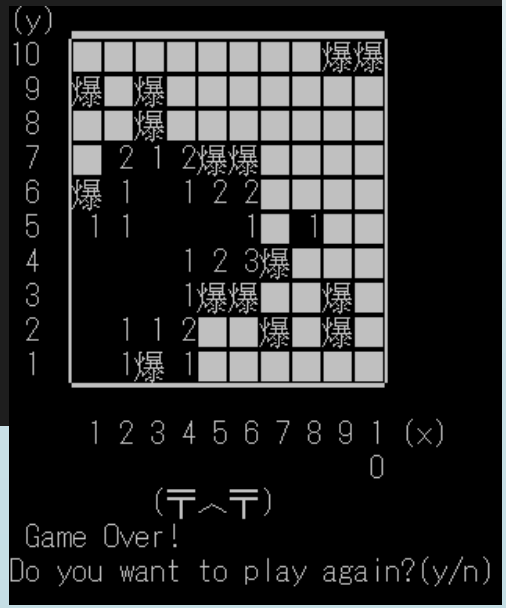
```
void NoMineAround(int x, int y){

    if (!map_press[x - 1][y - 1] && mine_info[x - 1][y - 1] != -2){//if (x - 1, y - 1) unstepped and is not a wall
        map_press[x - 1][y - 1] = 1;                //step (x - 1, y - 1)
        if (mine_info[x - 1][y - 1] == 0){// no mine around (x - 1, y - 1)
            NoMineAround(x - 1, y - 1);
        }
    }
    if (!map_press[x - 1][y] && mine_info[x - 1][y] != -2){
        map_press[x - 1][y] = 1;
        if (mine_info[x - 1][y] == 0){
            NoMineAround(x - 1, y);
        }
    }
    if (!map_press[x - 1][y + 1] && mine_info[x - 1][y + 1] != -2){
        map_press[x - 1][y + 1] = 1;
        if (mine_info[x - 1][y + 1] == 0){
            NoMineAround(x - 1, y + 1);
        }
    }
```

Flow chart


Start
Input length, mine_number
Is value legal?
no
yes
Init, and Set mine information
Show the chessboard
Input X,Y
Is X,Y legal?
No
Yes
Press the X,Y
Boom!
lose
Not Boom!
Some mines nearby
No
recursive
Yes
Hit the mine
The grids we traced = All grids – number of mines
No
win
Yes

Check() function

```
bool Check(){     //check win or lose
    int stepped_number = 0;
    for (int i = 1; i <= boardlength; i++){
        for (int j = 1; j <= boardlength; j++){
            if ((map_press[i][j])){
                stepped_number++;
                if (mine_info[i][j] == -1){ //step on the mine
                    for (int m = 1; m <= boardlength; m++){
                        for (int n = 1; n <= boardlength; n++){ //show all mines on the board
                            if (mine_info[m][n] == -1)
                                map_press[m][n] = 1;
                        }
                    }
                    ShowMineboard(0);
                    cout << " Game Over!\n";
                    return 0;
                }
            }
        }
    }
```

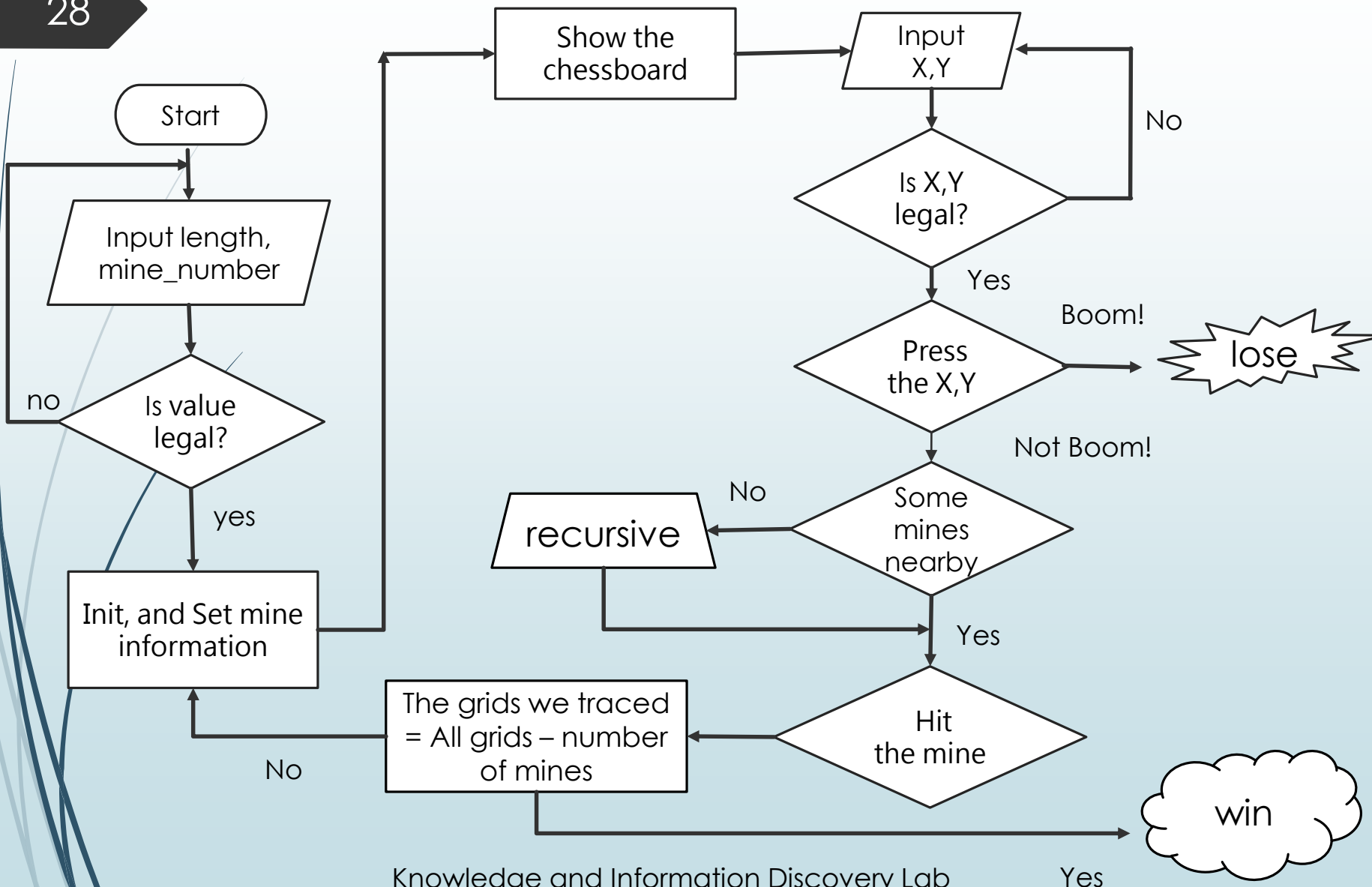


Check() function

```
171     /*Please compelete the following code.
172     You have to define the condition of win.
173     Also, you have to show the final result of the board
174     Hit: using the relation between stepped_ number and boardlength and mine_number and check all the mine_press array are all se
175     /*
176     if (){       //the condition of win
177         ShowMineboard(1);
178         cout << " Congratulations!!\n";
179         return 0;
180     }
181     else return 1; //game still continue
182     */
183     //5555555555555555555555555555555555555555555555555555test5555555555555555555555555
184
185 }
```

```
(y)
10    1 1 1   1雷 1
 9  1 2雷 1   1 1 1
 8 雷 2 1 1
 7  1 1
 6        1 1 1   1 1 1
 5      1 2雷 1   2雷 2
 4      2雷 3 1   2雷 2
 3      2雷 2 1 1 2 1 1
 2      1 1 1 2雷 2
 1            2雷 2

   1 2 3 4 5 6 7 8 9 1 (x)
                     0
        ~(￣▽￣)~
 Congratulations!!
Do you want to play again?(y/n)
```

Flow chart


Start
Input length, mine_number
Is value legal?
no
yes
Init, and Set mine information
Show the chessboard
Input X,Y
Is X,Y legal?
No
Yes
Press the X,Y
Boom!
lose
Not Boom!
Some mines nearby
No
recursive
Yes
Hit the mine
The grids we traced = All grids – number of mines
No
win
Yes