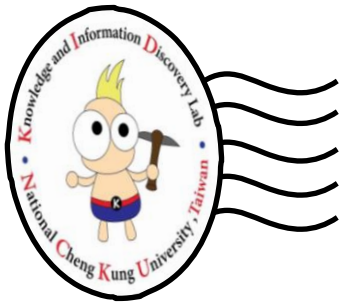


Minesweeper

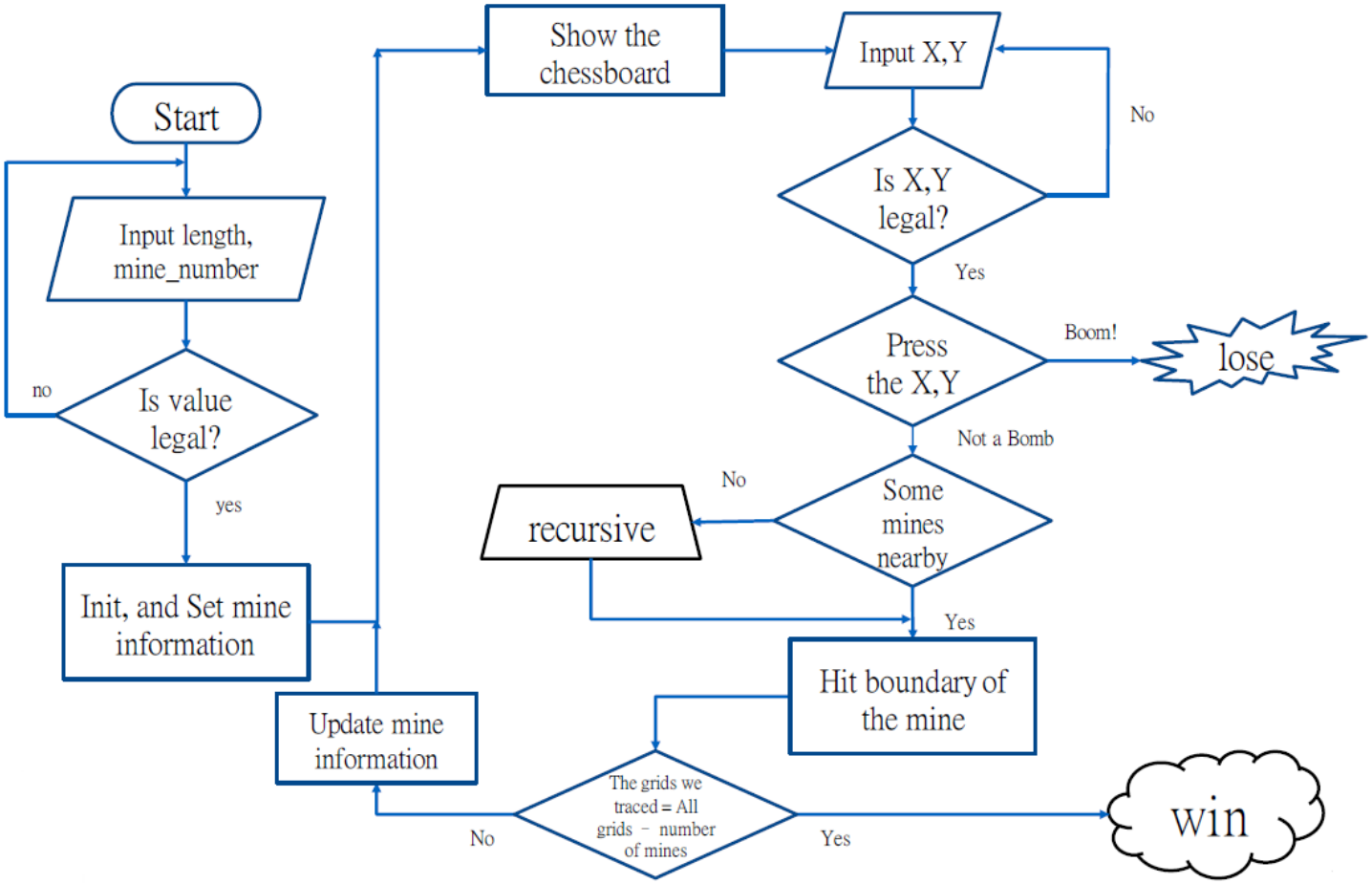
Speaker: Tzu-Hsin Yang

Advisor: Jen-Wei Huang

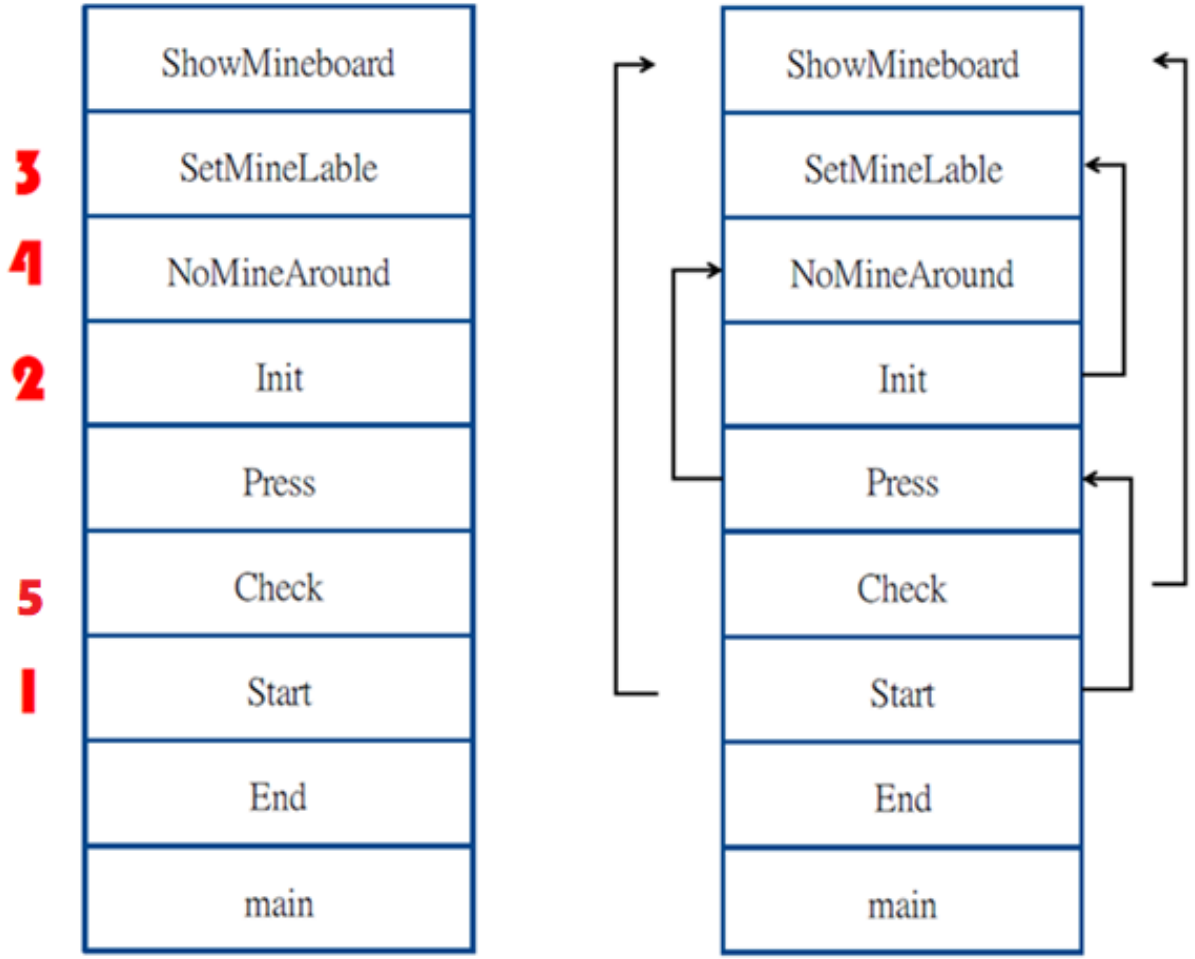
Aug. 18, 2017

Rules

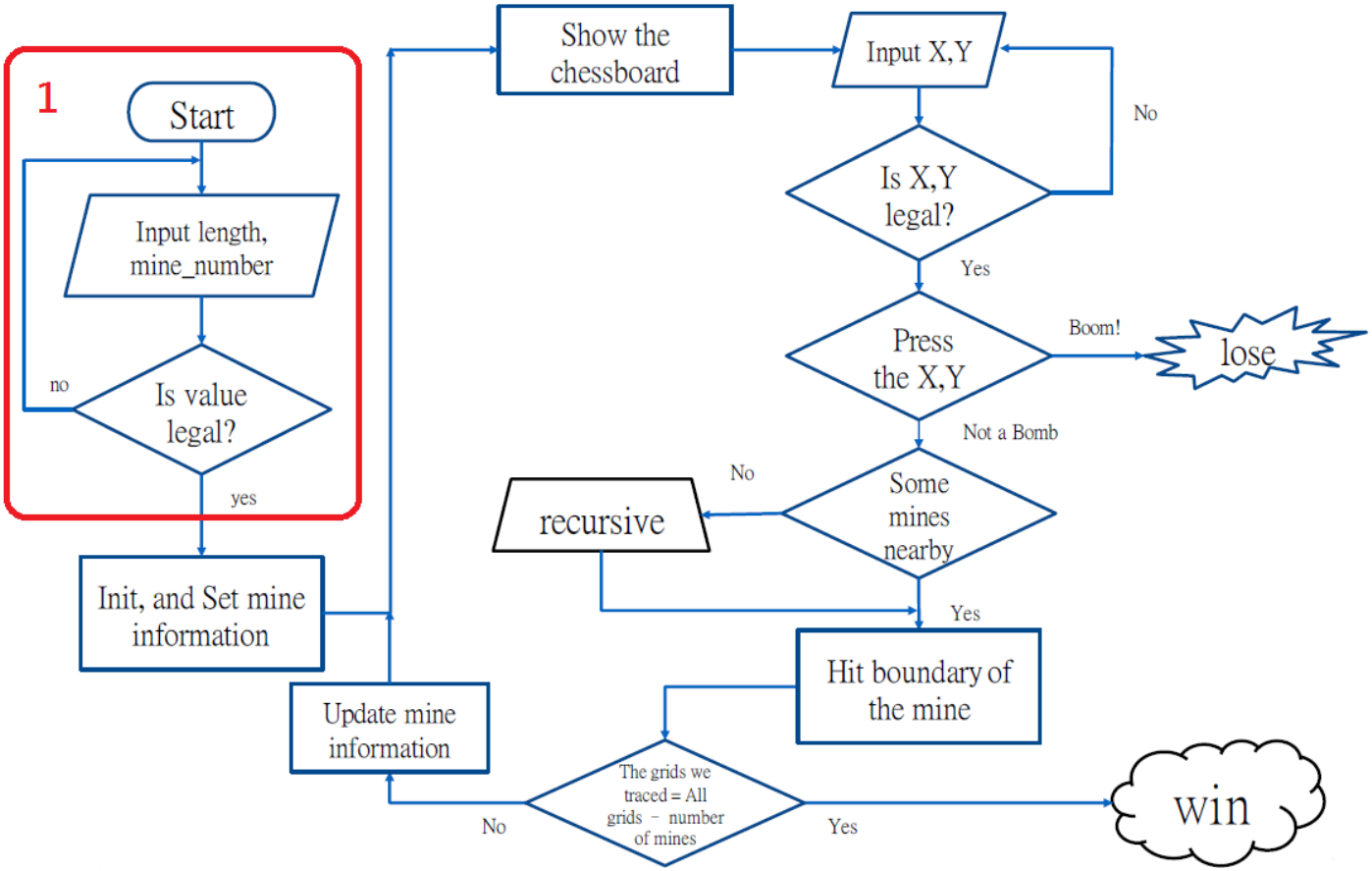
Flow Chart


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

Code Structure



Flow Chart


1
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 a Bomb
Some mines nearby
No
recursive
Yes
Hit boundary of the mine
Update mine information
The grids we traced = All grids - number of mines
No
Yes
win

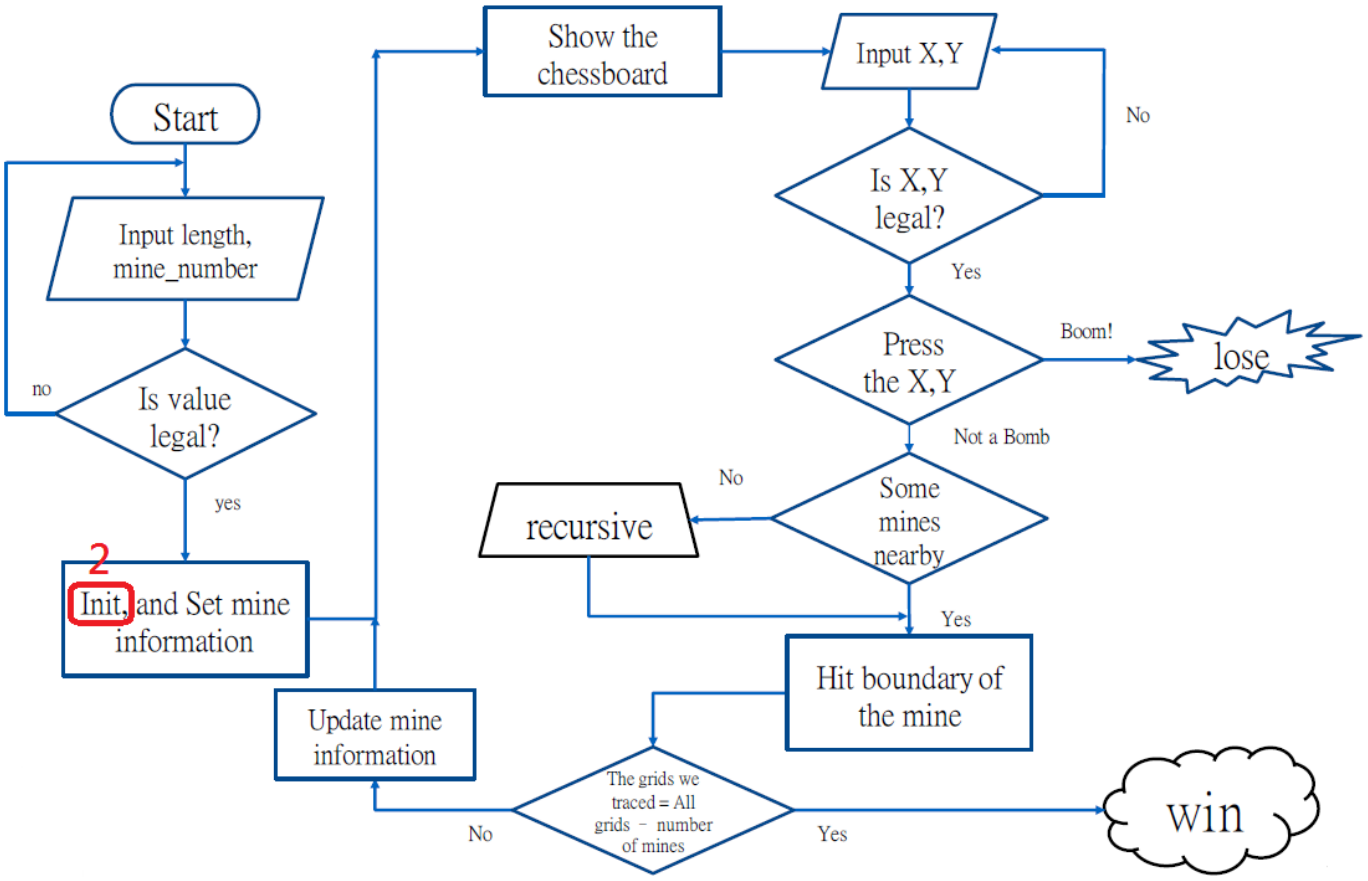
Start()

```
239 void Start(){           //start game
240     do{
241         input_error = 0;
242         cout << "Enter the board length(1~18)  :";
243         cin >> boardlength;
244         if(boardlength > boardlength_max || boardlength < 1)
245         input_error = 1;
246         if(input_error == 0){
247             cout << "Enter the mine number:" << "(" << "0~" << boardlength*boardlength << ")  :";
248             cin >> mine_number;
249             /*11111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111*/
250             /*Please compelete the following code.
251             Condition:
252                 When user's input value is illegal, then the process will ask user to retry.
253             Hint:
254                 using the if() and the argument "input_error" and use system("pause") to catch the screen.
255             Variable:
256                 mine_number, input_error
257             */
258             /*
259
260             if()// check conditions: if mine_number is smaller than zero or larger then board size, it is illegal
261             {
262               // show error message, set input_error, and pause screen
263             }
264             */
265             /*11111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111*/
266         }
267         system("cls");//clean screen
268     } while (input_error);
269     Init();
270     bool contiune = 1;
271     while (contiune){
272         ShowMineboard(2);
273         Press();
274         contiune = Check();
275     }
276 }
```

If input_error is set to 0 → input is legal

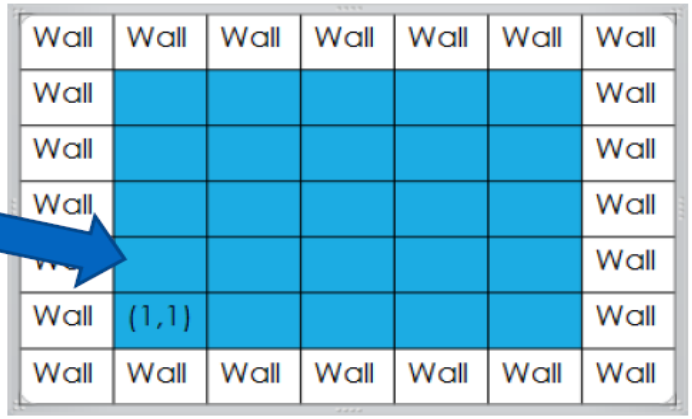
If input_error is set to 1 → input is illegal

Flow Chart


Start
Input length, mine_number
Is value legal?
no
yes
2
Init, and Set mine information
Update mine information
Show the chessboard
Input X,Y
Is X,Y legal?
No
Yes
Press the X,Y
Boom!
lose
Not a Bomb
Some mines nearby
No
recursive
Yes
Hit boundary of the mine
The grids we traced = All grids - number of mines
No
Yes
win

Init() - 1

(boardlength, boardlength)



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

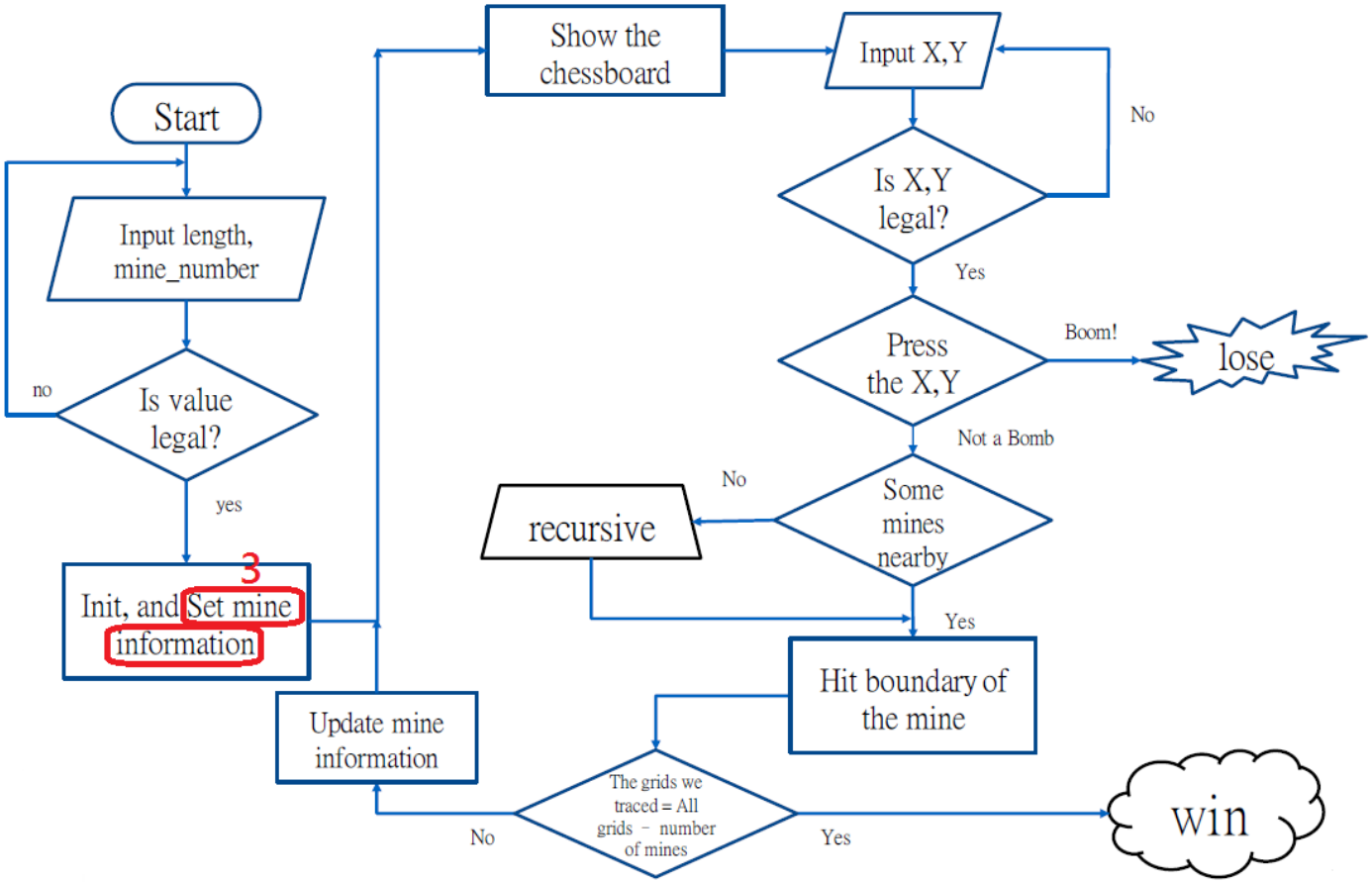
Init() - 2

```
123  void Init(){                                    //initialize
124      int counter = 0;
125      int i, j;
126      for (i = 1; i <= boardlength; i++){    //initialize the mineboard
127          for (j = 1; j <= boardlength; j++)
128          {
129              mine_info[i][j] = 0;
130              map_press[i][j] = 0;
131          }
132      }
133      for (i = 0; i <= boardlength + 1; i++){    //initialize the wall
134          mine_info[i][0] = -2;
135          mine_info[i][boardlength + 1] = -2;
136      }
137      for (j = 0; j <= boardlength + 1; j++){    //initialize the wall
138          mine_info[0][j] = -2;
139          mine_info[boardlength + 1][j] = -2;
140      }
```

Init() - 3

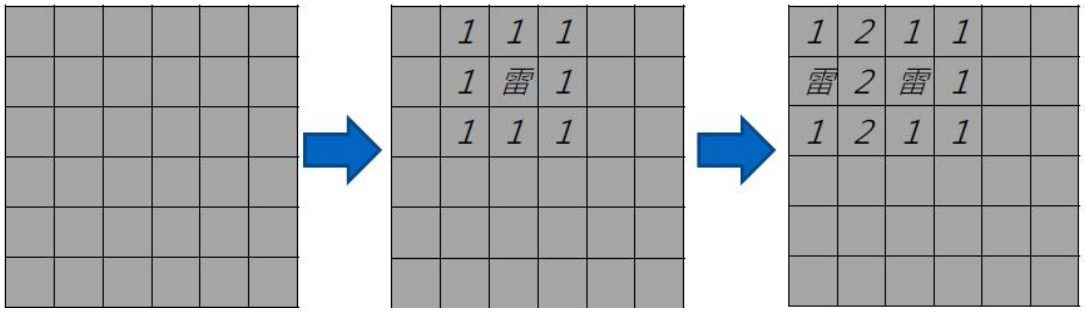
```
141     srand(time(NULL));
142     /*2222222222222222222222222222222222222222222222222222222222222222*/
143     /*
144     Please compelete the following code.
145     Set n mines randomly on location (x, y), n = mine_number.
146     Let x and y belong to 1 ~ boardlength.
147     Hint:
148         using the argument "counter" and %.
149         mine_info[][] can express the mine.
150     Variable:
151         mine_number, mine_info[][], counter, x, y.
152     */
153
154     /*
155     while ()      //control the mumber of mines
156     {
157         int x = rand();
158         int y = rand();
159         if ()    //if no mine at this location, then put the mine in
160         {
161
162             SetMineLable(x, y);  //do not change this line!!
163
164         }
165     }
166     */
167     /*2222222222222222222222222222222222222222222222222222222222222222*/
168 }
```

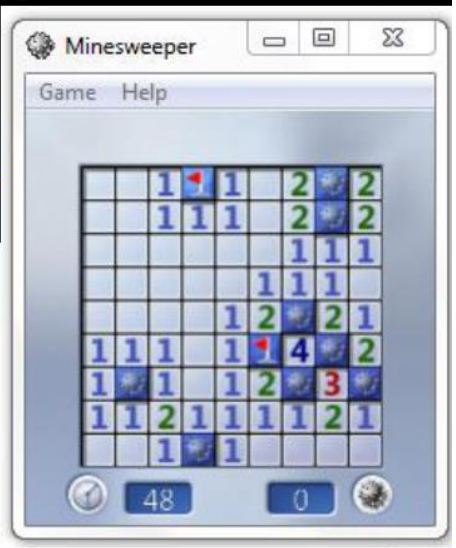
Flow Chart


Start
Input length, mine_number
Is value legal?
no
yes
Init, and Set mine information
3
Show the chessboard
Input X,Y
Is X,Y legal?
No
Yes
Press the X,Y
Boom!
lose
Not a Bomb
Some mines nearby
No
recursive
Yes
Hit boundary of the mine
The grids we traced = All grids - number of mines
No
Yes
win
Update mine information

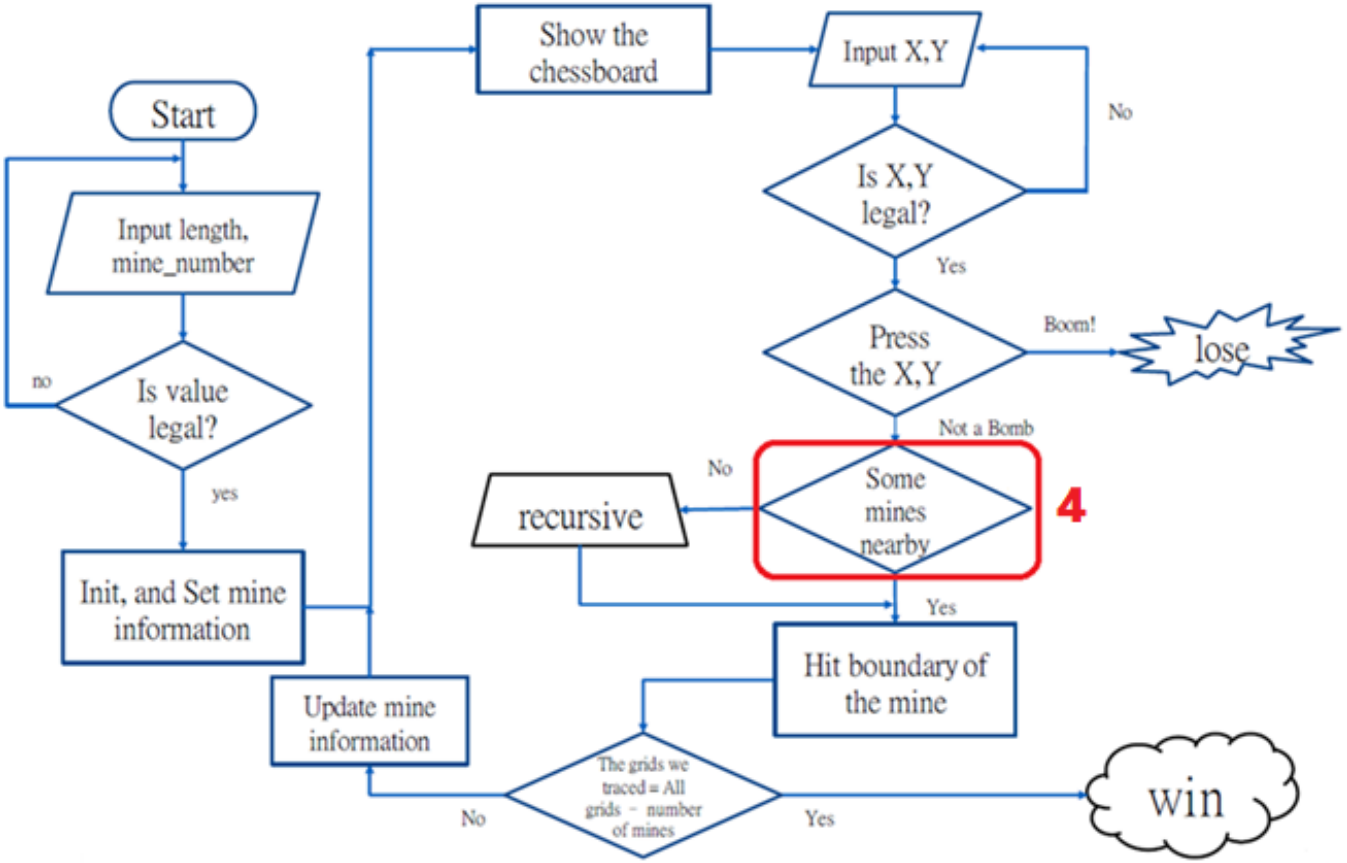
SetMineLabel()

```
71  void SetMineLable(int x, int y) //update how many mines around
72  {
73      //3333333333333333333333test33333333333333333333
74      /* check 8 location that around the (x, y),
75      if the location is not mine and not wall then update the mine number
76      Hit: using the mine_info[][]
77      */
78
79
80      //3333333333333333333333test33333333333333333333
81  }
```



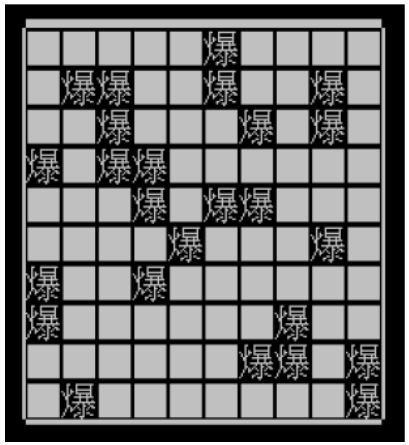


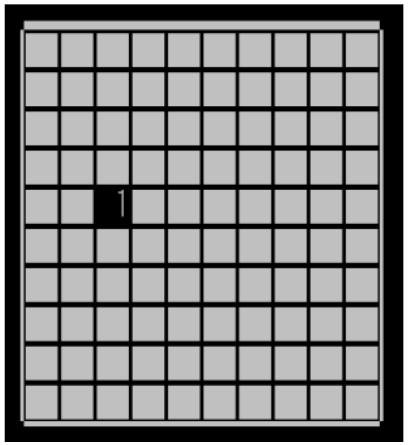
Flow Chart

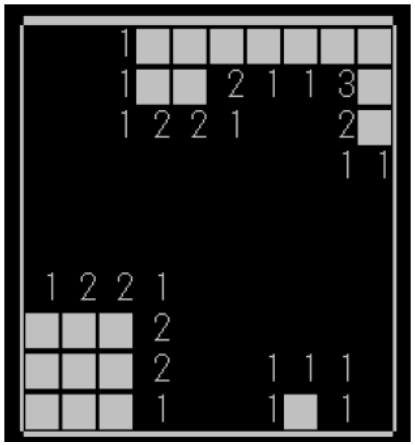

Start
Input length, mine_number
Is value legal?
no
yes
Init, and Set mine information
Show the chessboard
Input X,Y
Is X,Y legal?
No
Yes
Press the X,Y
Boom!
lose
Not a Bomb
Some mines nearby
No
recursive
4
Yes
Hit boundary of the mine
The grids we traced = All grids - number of mines
No
Yes
win
Update mine information

Three Case when Press

- Hit a Bomb
- No explosion, but there are bombs nearby
- No explosion, no bomb nearby

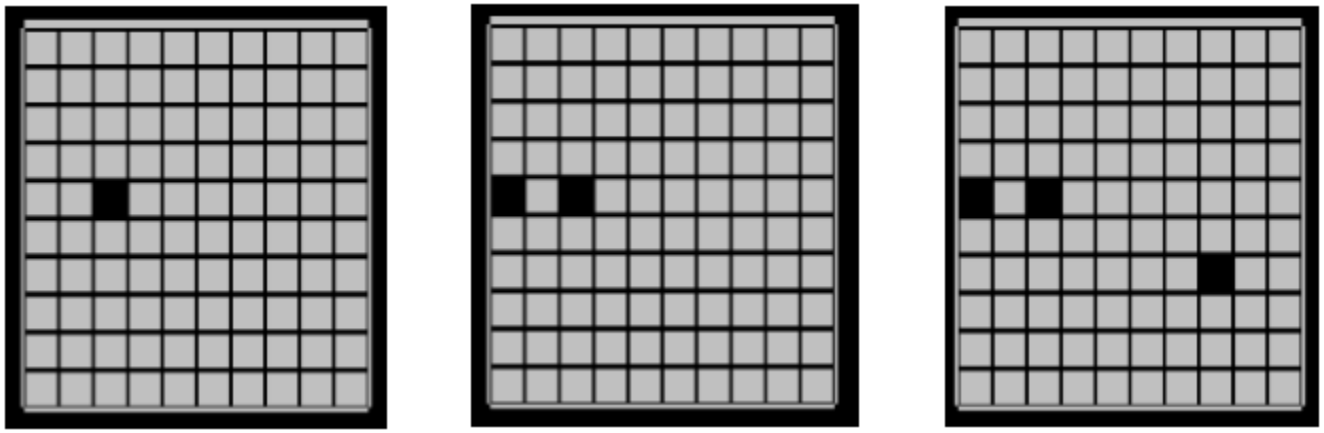




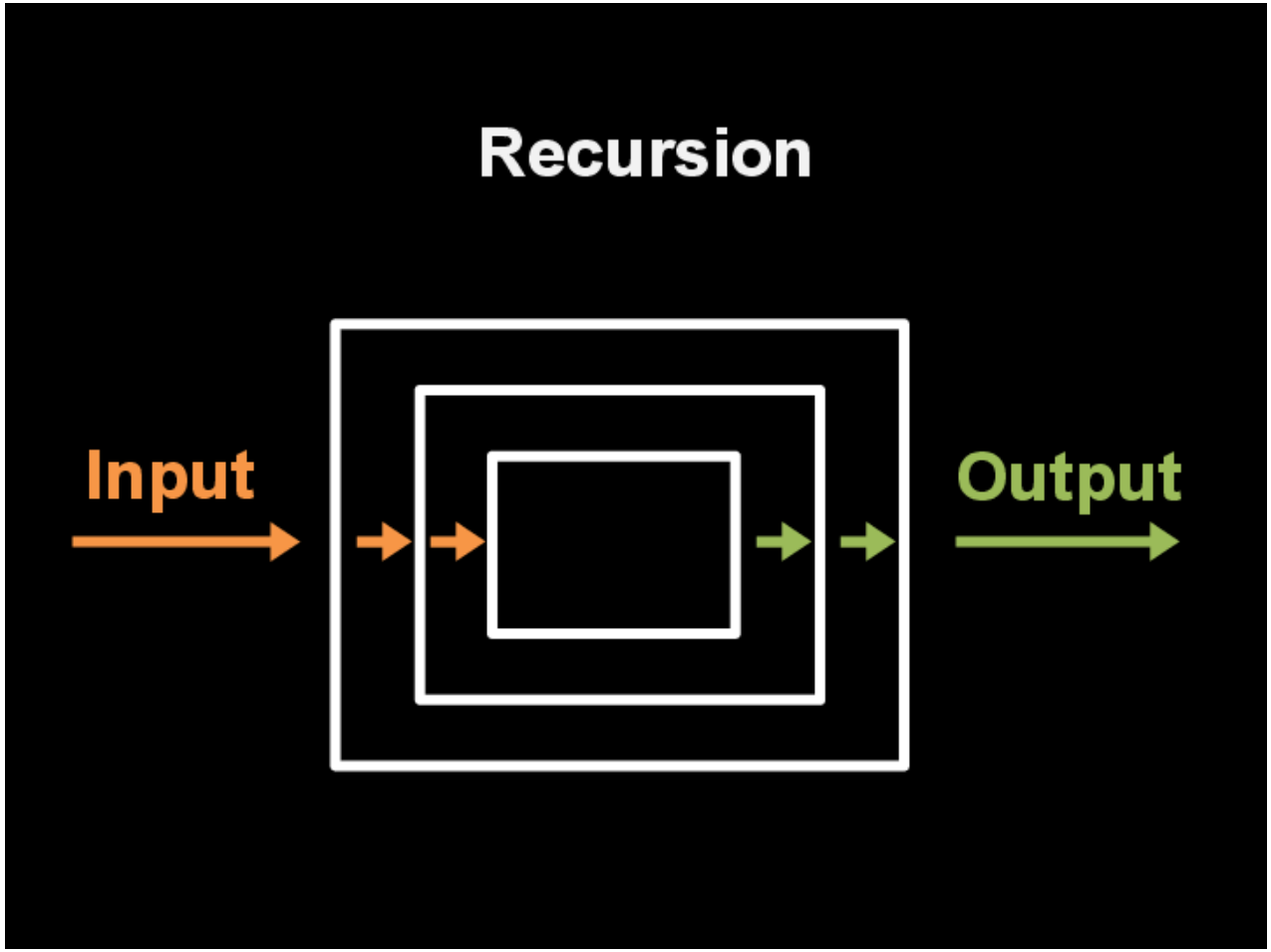


No Bombs nearby

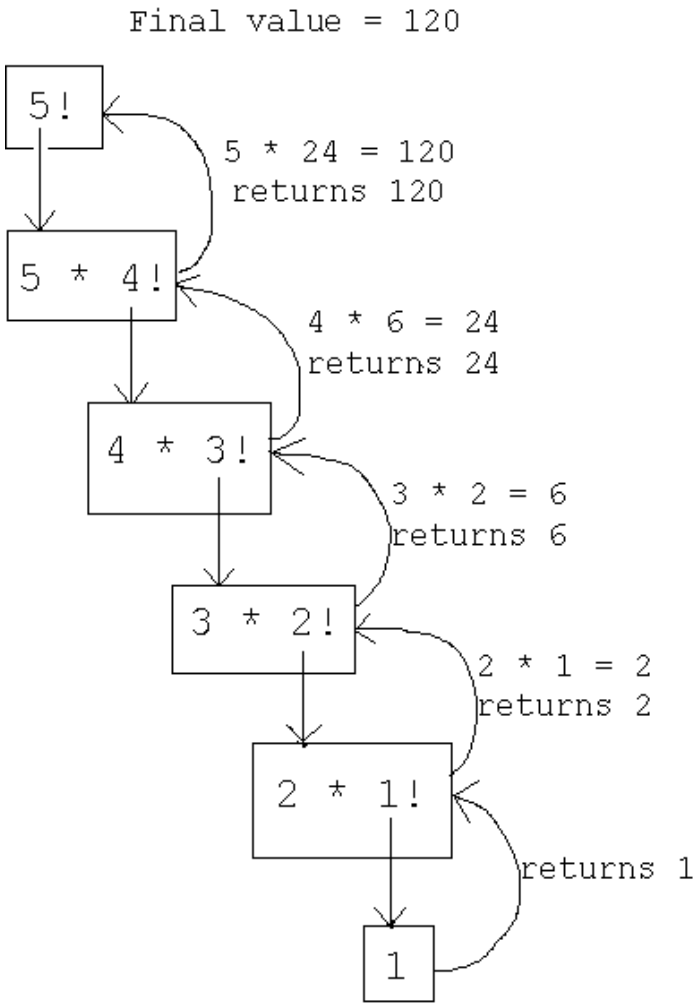
- Press one cell at a time



Recursion



Simple Example




```
int factorial(int n){
    if(n == 1){
        return 1;
    }
    return n * factorial(n-1);
}
```

```
factorial(N);
```

Without Return Value

```
void function(int a, int b){
    if(/*condition*/){
        //call itself
        function(a+1, b+1);
    }
}

function(x, y)
```

Without Return Value

```
void function(int a, int b){
    if(/*condition*/){
        //call itself
        function(a+1, b+1);
    }
}

function(x, y)
```

Without Return Value

```
void function(int a, int b){
    if(/*condition*/){
        //call itself
        function(a+1, b+1);
    }
}

function(x, y)
```

Without Return Value

```
void function(int a, int b){
    if(/*condition*/){
        //call itself
        function(a+1, b+1);
    }
}

function(x, y)
```

Without Return Value

```
void function(int a, int b){
    if(/*condition*/){
        //call itself
        function(a+1, b+1);
    }
}

function(x, y)
```

Without Return Value

```
void function(int a, int b){
    if(/*condition*/){
        //call itself
        function(a+1, b+1);
    }
}

function(x, y)
```

Without Return Value

```
void function(int a, int b){
    if(/*condition*/){
        //call itself
        function(a+1, b+1);
    }
}

function(x, y)
```

Without Return Value

```
void function(int a, int b){
    if(/*condition*/){
        //call itself
        function(a+1, b+1);
    }
}

function(x, y)
```

Without Return Value

```
void function(int a, int b){
    if(/*condition*/){
        //call itself
        function(a+1, b+1);
    }
}

function(x, y)
```

……

NoMineAround()

```
166 void Press(){    //step
167     input_error = 0;
168     cout << "input x:  ";
169     cin >> x;
170     cout << "input y:  ";
171     cin >> y;
172     system("cls");
173     if (0 < x && x <= boardlength && 0 < y && y <= boardlength)
174         map_press[x][y] = 1;//step (x, y)
175     else{
176         input_error = 1;
177         cout << "input error, please try again!" << "\n";
178         system("pause");
179         system("cls");
180     }
181     if (input_error == 0 && mine_info[x][y] == 0) NoMineAround(x, y);//if there is no mine on (x, y) , call NoMineAround()
182 }
```

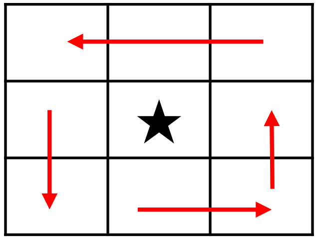
Call the function “NoMineAround” here

NoMineAround()

```
92  void NoMineAround(int x, int y){
93      /*4444444444444444444444444444444444444444444444444444444444444444444444444444444444*/
94      /*
95      If the location is not press and it is not a wall,
96      First we set the location to "press",
97      Then we check the mine_info if it is zero,
98      Then we run the NoMineAround again (Recursive) to travel all blank cells
99
100     Variable:
101         map_press[][], mine_info[][], x, y
102
103     Hint:
104         Need to check 8 directions and itself
105     */
106
107     /*
108     if (){      //if (x - 1, y - 1) unstepped and is not a wall
109                 //step (x - 1, y - 1)
110         if (){  // no mine around (x - 1, y - 1)
111                 // go in (x - 1, y - 1) to check
112         }
113     }
114     */
115
116     /*4444444444444444444444444444444444444444444444444444444444444444444444444444444444*/
117 }
```

Every location except mine should set to “press” for final success requirement

Recursion



	1	1	1					
	1	雷	1	1	2	3	2	1
	1	1	1	1	雷	雷	雷	2
			1	2	3	4	雷	2
1	1		1	雷	1	1	1	1
雷	1		1	2	3	2	1	
1	2	1	1	1	雷	雷	1	
	1	雷	1	1	2	2	1	

	1	1	1					
	1	雷	1	1	2	3	2	1
	1	1	1	1	雷	雷	雷	2
		★	1	2	3	4	雷	2
1	1		1	雷	1	1	1	1
雷	1		1	2	3	2	1	
1	2	1	1	1	雷	雷	1	
	1	雷	1	1	2	2	1	

	1	1	1					
	1	雷	1	1	2	3	2	1
	1	1	1	1	雷	雷	雷	2
		★	1	2	3	4	雷	2
1	1		1	雷	1	1	1	1
雷	1		1	2	3	2	1	
1	2	1	1	1	雷	雷	1	
	1	雷	1	1	2	2	1	

	1	1	1					
	1	雷	1	1	2	3	2	1
	1	1	1	1	雷	雷	雷	2
		★	1	2	3	4	雷	2
1	1		1	雷	1	1	1	1
雷	1		1	2	3	2	1	
1	2	1	1	1	雷	雷	1	
	1	雷	1	1	2	2	1	

	1	1	1					
	1	雷	1	1	2	3	2	1
	1	1	1	1	雷	雷	雷	2
		★	1	2	3	4	雷	2
1	1		1	雷	1	1	1	1
雷	1		1	2	3	2	1	
1	2	1	1	1	雷	雷	1	
	1	雷	1	1	2	2	1	

	1	1	1					
	1	雷	1	1	2	3	2	1
	1	1	1	1	雷	雷	雷	2
	★		1	2	3	4	雷	2
1	1		1	雷	1	1	1	1
雷	1		1	2	3	2	1	
1	2	1	1	1	雷	雷	1	
	1	雷	1	1	2	2	1	

Recursion

	1	1	1					
	1	雷	1	1	2	3	2	1
	1	1	1	1	雷	雷	雷	2
	★		1	2	3	4	雷	2
1	1		1	雷	1	1	1	1
雷	1		1	2	3	2	1	
1	2	1	1	1	雷	雷	1	
	1	雷	1	1	2	2	1	

	1	1	1					
	1	雷	1	1	2	3	2	1
★	1	1	1	1	雷	雷	雷	2
			1	2	3	4	雷	2
1	1		1	雷	1	1	1	1
雷	1		1	2	3	2	1	
1	2	1	1	1	雷	雷	1	
	1	雷	1	1	2	2	1	

	1	1	1					
	1	雷	1	1	2	3	2	1
★	1	1	1	1	雷	雷	雷	2
			1	2	3	4	雷	2
1	1		1	雷	1	1	1	1
雷	1		1	2	3	2	1	
1	2	1	1	1	雷	雷	1	
	1	雷	1	1	2	2	1	

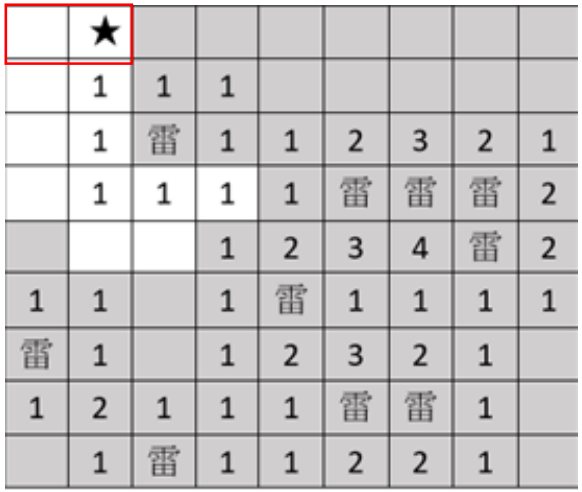
No way to go!

★	1	1	1					
	1	雷	1	1	2	3	2	1
	1	1	1	1	雷	雷	雷	2
			1	2	3	4	雷	2
1	1		1	雷	1	1	1	1
雷	1		1	2	3	2	1	
1	2	1	1	1	雷	雷	1	
	1	雷	1	1	2	2	1	

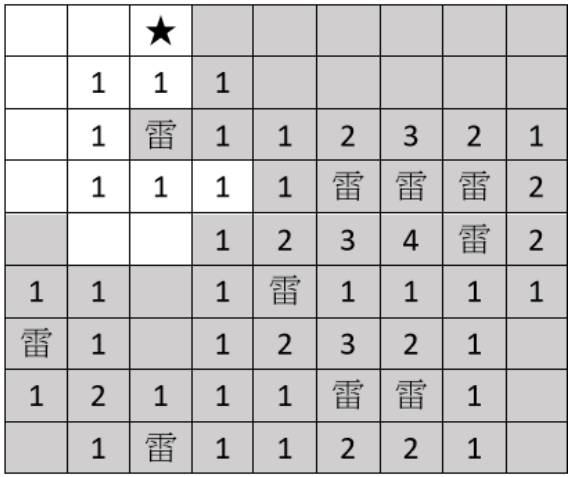
	★							
	1	1	1					
	1	雷	1	1	2	3	2	1
	1	1	1	1	雷	雷	雷	2
			1	2	3	4	雷	2
1	1		1	雷	1	1	1	1
雷	1		1	2	3	2	1	
1	2	1	1	1	雷	雷	1	
	1	雷	1	1	2	2	1	

★								
	1	1	1					
	1	雷	1	1	2	3	2	1
	1	1	1	1	雷	雷	雷	2
			1	2	3	4	雷	2
1	1		1	雷	1	1	1	1
雷	1		1	2	3	2	1	
1	2	1	1	1	雷	雷	1	
	1	雷	1	1	2	2	1	

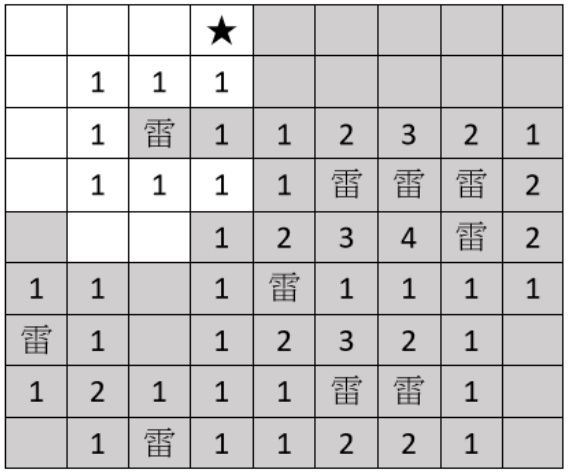
Recursion



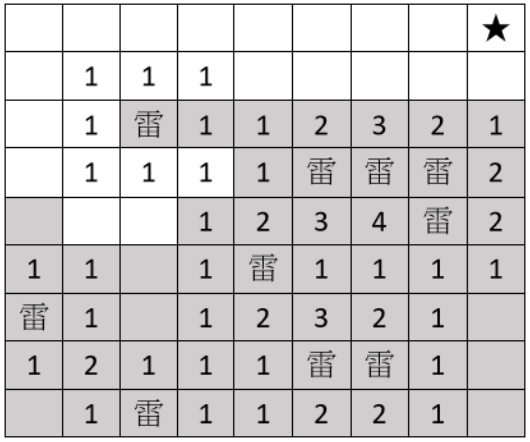
	★							
	1	1	1					
	1	雷	1	1	2	3	2	1
	1	1	1	1	雷	雷	雷	2
			1	2	3	4	雷	2
1	1		1	雷	1	1	1	1
雷	1		1	2	3	2	1	
1	2	1	1	1	雷	雷	1	
	1	雷	1	1	2	2	1	



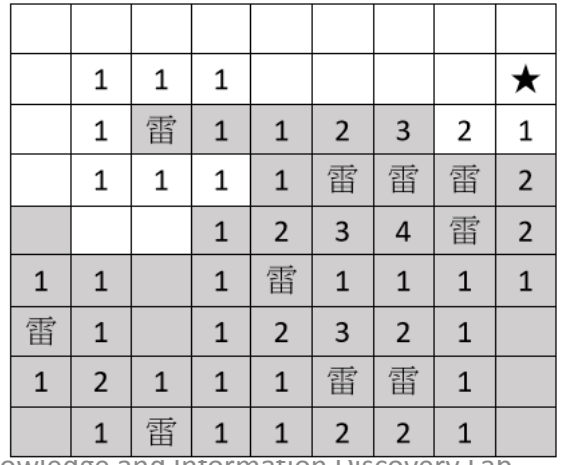
		★						
	1	1	1					
	1	雷	1	1	2	3	2	1
	1	1	1	1	雷	雷	雷	2
			1	2	3	4	雷	2
1	1		1	雷	1	1	1	1
雷	1		1	2	3	2	1	
1	2	1	1	1	雷	雷	1	
	1	雷	1	1	2	2	1	



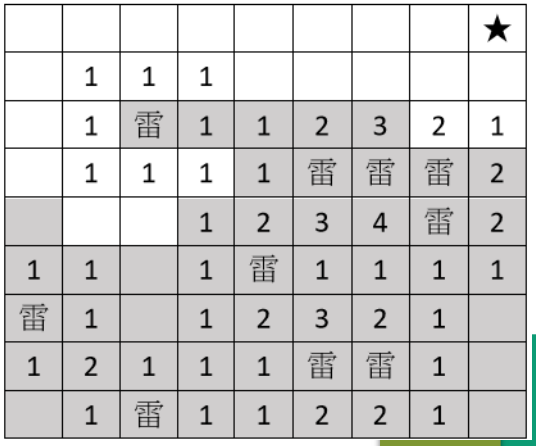
			★					
	1	1	1					
	1	雷	1	1	2	3	2	1
	1	1	1	1	雷	雷	雷	2
			1	2	3	4	雷	2
1	1		1	雷	1	1	1	1
雷	1		1	2	3	2	1	
1	2	1	1	1	雷	雷	1	
	1	雷	1	1	2	2	1	



								★
	1	1	1					
	1	雷	1	1	2	3	2	1
	1	1	1	1	雷	雷	雷	2
			1	2	3	4	雷	2
1	1		1	雷	1	1	1	1
雷	1		1	2	3	2	1	
1	2	1	1	1	雷	雷	1	
	1	雷	1	1	2	2	1	



	1	1	1					★
	1	雷	1	1	2	3	2	1
	1	1	1	1	雷	雷	雷	2
			1	2	3	4	雷	2
1	1		1	雷	1	1	1	1
雷	1		1	2	3	2	1	
1	2	1	1	1	雷	雷	1	
	1	雷	1	1	2	2	1	



								★
	1	1	1					
	1	雷	1	1	2	3	2	1
	1	1	1	1	雷	雷	雷	2
			1	2	3	4	雷	2
1	1		1	雷	1	1	1	1
雷	1		1	2	3	2	1	
1	2	1	1	1	雷	雷	1	
	1	雷	1	1	2	2	1	

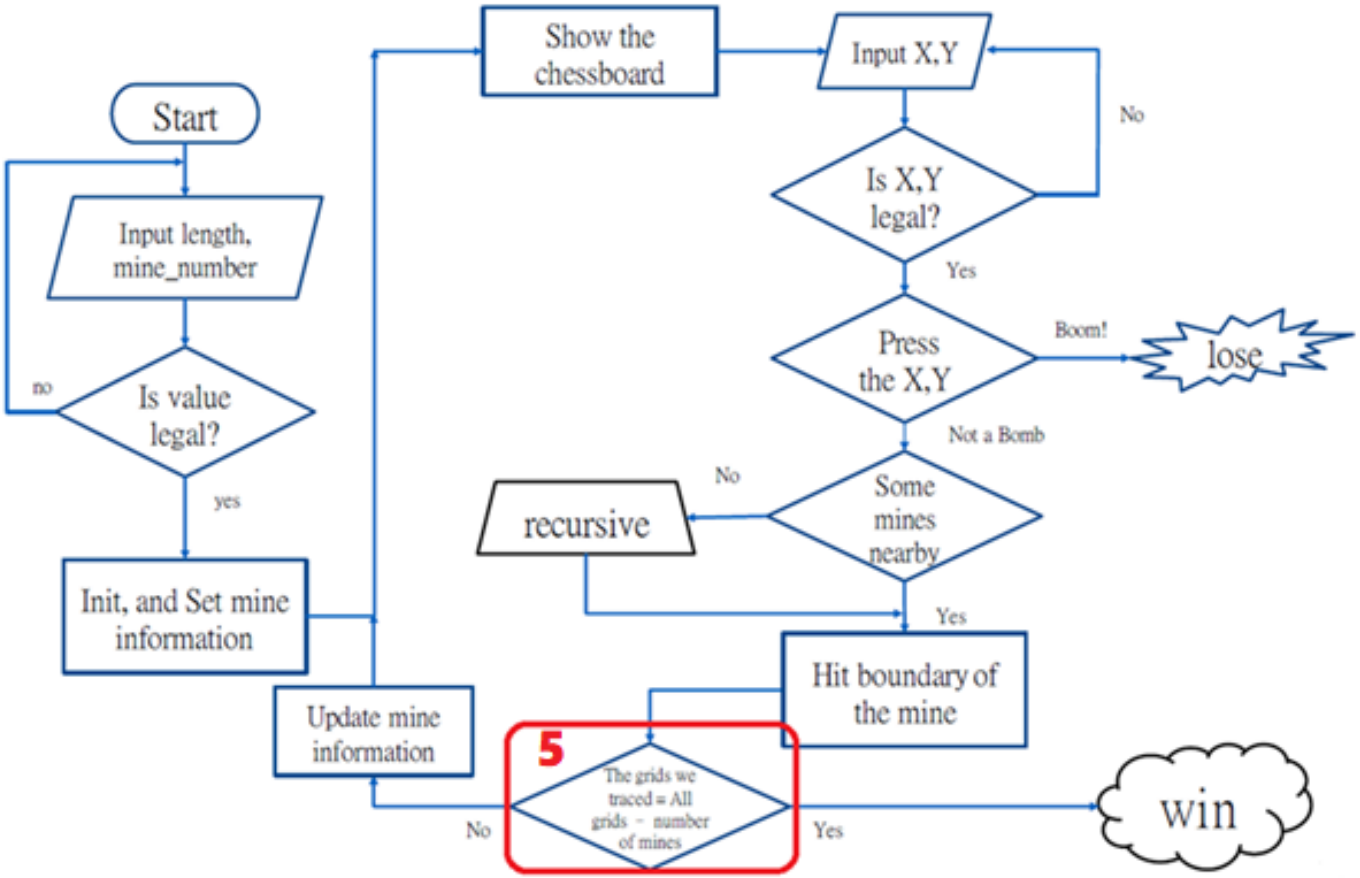
Recursion

	1	1	1					
	1	雷	1	1	2	3	2	1
	1	1	1	1	雷	雷	雷	2
			1	2	3	4	雷	2
1	1		1	雷	1	1	1	1
雷	1	★	1	2	3	2	1	
1	2	1	1	1	雷	雷	1	
	1	雷	1	1	2	2	1	

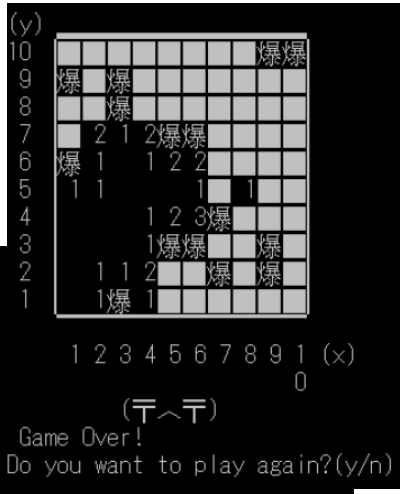
NoMineAround()

```
92  void NoMineAround(int x, int y){
93      /*4444444444444444444444444444444444444444444444444444444444444444444444444444444444*/
94      /*
95      If the location is not press and it is not a wall,
96      First we set the location to "press",
97      Then we check the mine_info if it is zero,
98      Then we run the NoMineAround again (Recursive) to travel all blank cells
99
100     Variable:
101         map_press[][], mine_info[][], x, y
102
103     Hint:
104         Need to check 8 directions and itself
105     */
106
107     /*
108     if (){      //if (x - 1, y - 1) unstepped and is not a wall
109                 //step (x - 1, y - 1)
110         if (){  // no mine around (x - 1, y - 1)
111                 // go in (x - 1, y - 1) to check
112         }
113     }
114     */
115
116     /*4444444444444444444444444444444444444444444444444444444444444444444444444444444444*/
117 }
```

Flow Chart


Start
Input length, mine_number
Is value legal?
no
yes
Init, and Set mine information
Update mine information
Show the chessboard
Input X,Y
Is X,Y legal?
No
Yes
Press the X,Y
Boom!
lose
Not a Bomb
Some mines nearby
No
recursive
Yes
Hit boundary of the mine
5
The grids we traced = All grids - number of mines
No
Yes
win

Check() - 1




```
188  bool Check(){      //check win or lose
189      int stepped_number = 0;      //已踩過的格子總數
190      for (int i = 1; i <= boardlength; i++){
191          for (int j = 1; j <= boardlength; j++){
192              if ((map_press[i][j])){
193                  stepped_number++;
194                  if (mine_info[i][j] == -1){ //踩到地雷
195                      for (int m = 1; m <= boardlength; m++){
196                          for (int n = 1; n <= boardlength; n++){ //show所有地雷位置
197                              if (mine_info[m][n] == -1)
198                                  map_press[m][n] = 1;
199                          }
200                      }
201                      ShowMineboard(0);
202                      cout << " Game Over!\n";
203                      return 0;
204                  }
205              }
206          }
207      }
```

Check() - 2

```
/*5555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555*/
/*
Please compelete the following code.
You have to define the condition of win.
Also, you have to show the final result of the board

Hint:
    using the relation between stepped_ number and boardlength and mine_number and check all the mine_press array are all set to 1
Variable:
    stepped_number, boardlength, mine_number, map_press[][]

*/

/*
if ()           //the condition of win that whole map is pressed
{
    //Don't change code below
    for (int i = 1; i <= boardlength; i++){
        for (int j = 1; j <= boardlength; j++){  //view all
            map_press[i][j] = 1;
        }
    }
    ShowMineboard(1);
    cout << " Congratulations!!\n";
    return 0;
}
else return 1; //game still continue
*/
/*5555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555*/
```

```
(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)
```