CSCI 1540 Fundamental Computing with C++

Tutorial 10

Xiaopeng Zhang

SHB 913

xpzhang@cse.cuhk.edu.hk

Outline

- Assignment 5: Game of Hex

Requirement

- Assignment 5: Game of Hex
- Deadline: 20:00, Thu 21 Nov 2019
- Requirements:
  - Filename: hex.cpp;
  - Insert your name, student ID, and e-mail address as comments at the beginning of your source file;
  - The output must exactly match the sample output;
  - Include suitable comments as documentation;
  - Free of compilation errors and warnings;
  - You cannot declare any global variables (except the const ones);
  - At least four functions (including main());
    At least two functions should have array parameter(s).

Game Description

- Start condition:
  - There are 2 players (player O and player X) and an empty diamond game board.
- Game stage:
  - The players take turn to place their pieces on empty cells of the board. *(Player O takes the first turn.)*
- End condition:
  - Player O's winning condition: form a chain of his/her pieces connecting the **left** and **right** border of the board.
  - Player X's winning condition: form a chain of his/her pieces connecting the **top** and **bottom** border of the board.

Game Description

```
 A B C D E F G H I J K
0 . . . . . . . . . . .
 1 . . . . . . . . . . .
  2 . . . . . . . . . . .
   3 . . . . . . . . . . .
    4 . . . . . . . . . . .
     5 . . . . . . . . . . .
      6 . . . . . . . . . . .
       7 . . . . . . . . . . .
        8 . . . . . . . . . . .
         9 . . . . . . . . . . .
         10 . . . . . . . . . . .
```

Start condition

→

```
 A B C D E F G H I J K
0 O . . . . . . . . . .
 1 . . . . . . . . . . .
  2 . . . . . . . . . . .
   3 . . . . . . . . . . .
    4 . . . . . . . . . . .
     5 . . . . . . . . . . .
      6 . . . . . . . . . . .
       7 . . . . . . . . . . .
        8 . . . . . . . . . . .
         9 . . . . . . . . . . .
         10 . . . . . . . . . . .
```

Player O places a piece

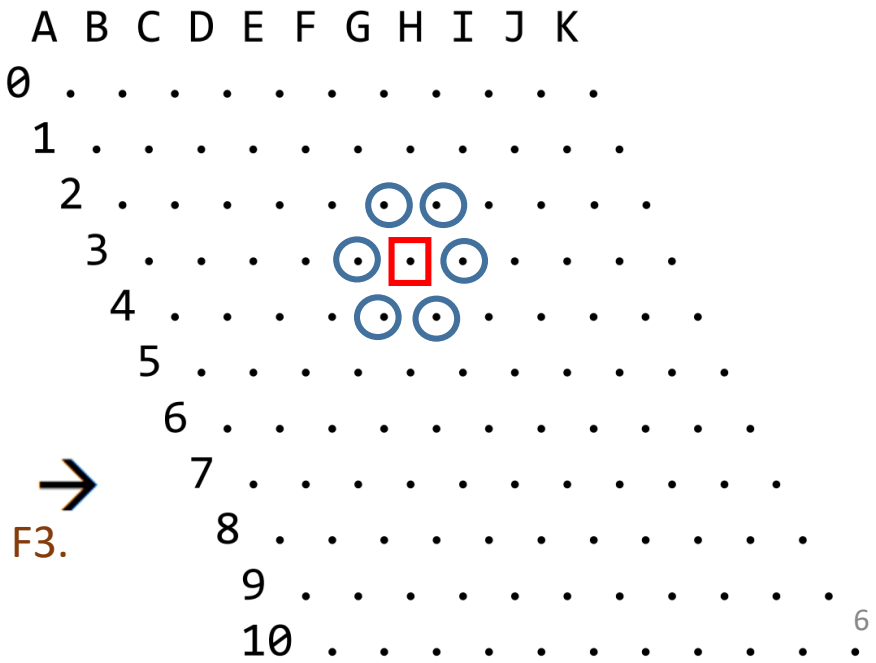
→ … →

```
 A B C D E F G H I J K
0 O X O O X O O X X X X
 1 O O X X O X O X X X O
  2 X O X O X X O X X O X
   3 O O O X O O O . O O O
    4 X O O O . X X X O X X
     5 X . X O O O O X O O X
      6 O O X O O X O O . O O
       7 . X . X X O O O X O X
        8 X X X O X X X O X O O
         9 X X X X X O O O X X X
         10 O X X O X X O O X O O
```

End condition

Board Representation

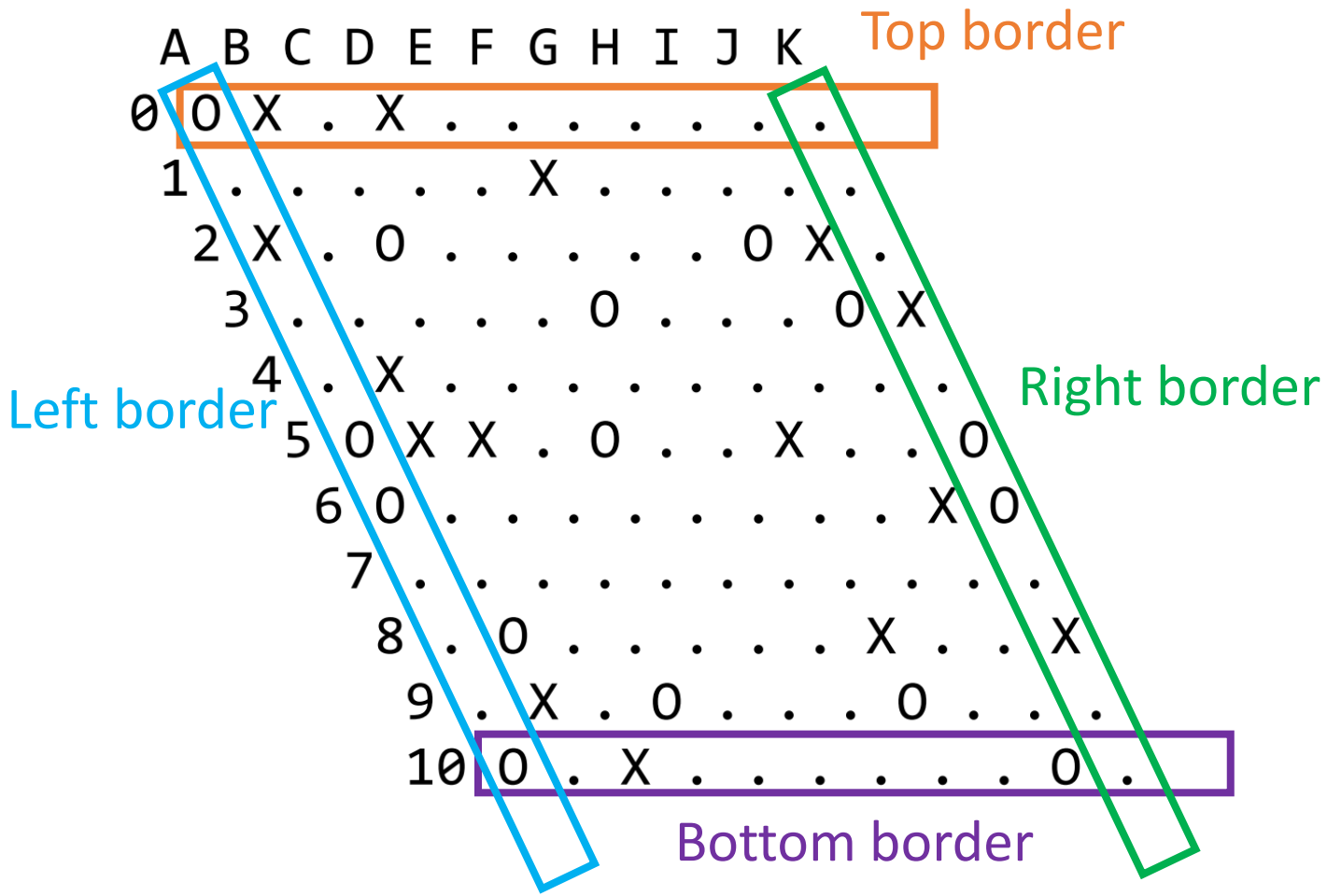
- There are 11 * 11 cells in a board.
- In each cell, “O” means the piece of player O; “X” means the piece of player X; and “.” means an empty cell.
- The rows and columns are named in numbers **(0–10)** and letters **(A–K)** respectively.
- Each cell has at most **six neighbors**.
  - For eamaple: cell F3 has neighbors F2, G2, E3, G3, E4, and F4. (E2 and G4 are not neighbors of F3.)



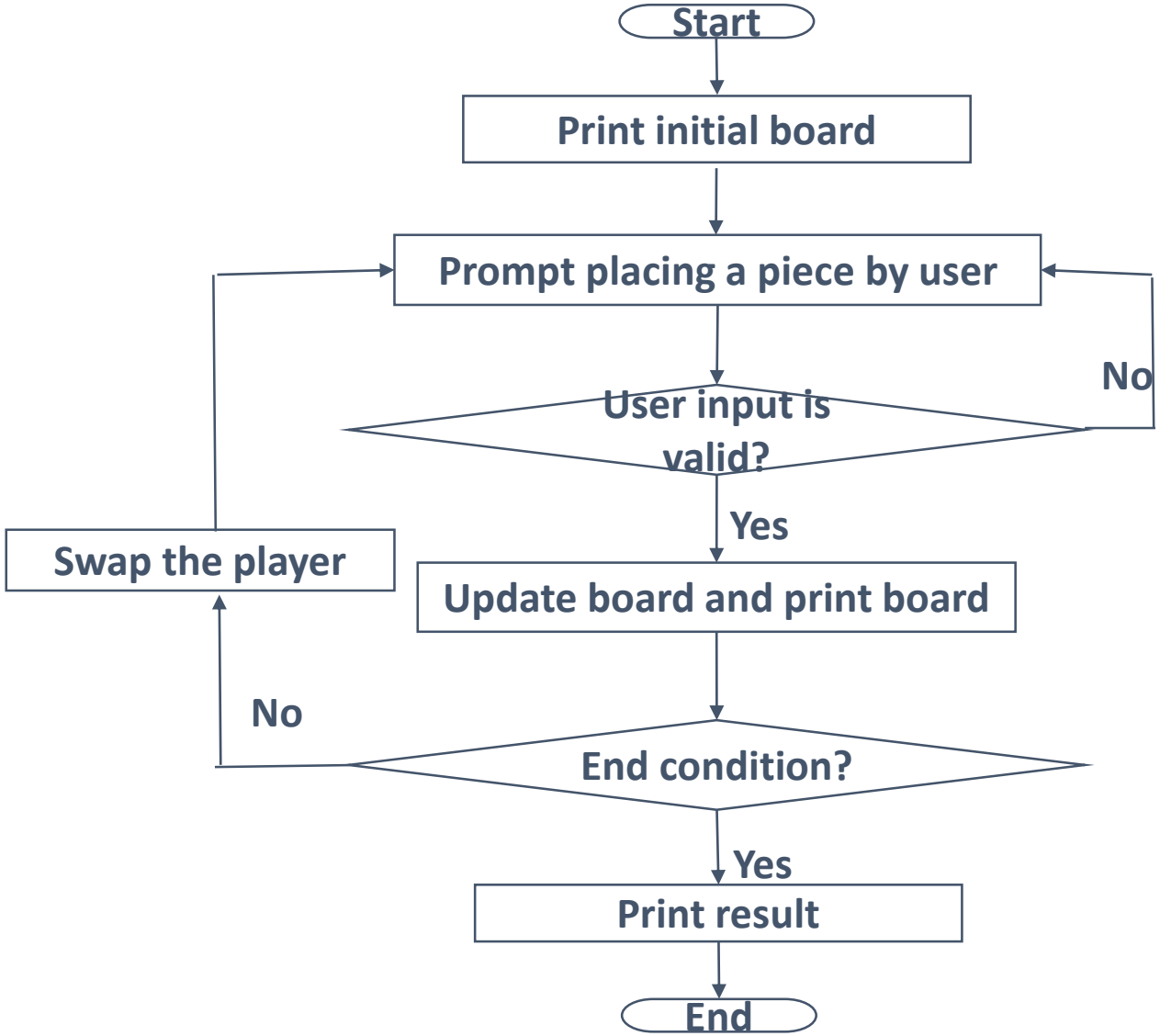
□ is F3;
○ means the neighbors of F3.

Board Representation

- Borders:



Program Flow



Possible Functions

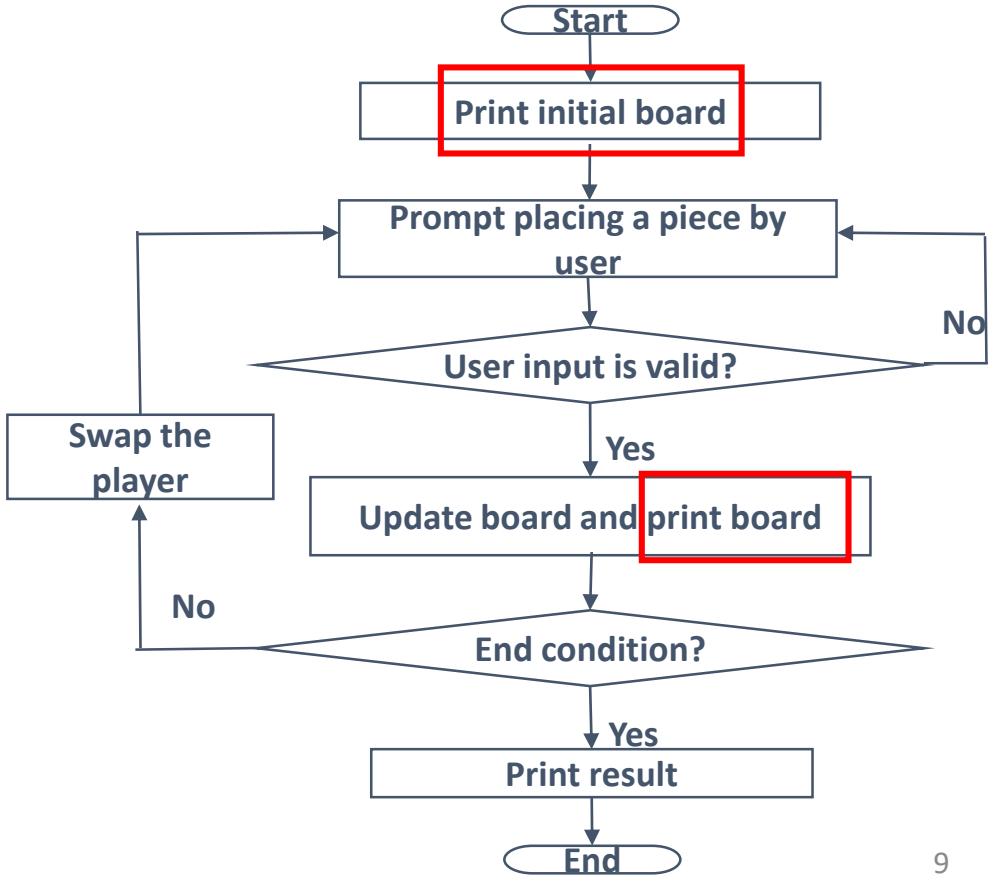
- A function prints the board to the screen using the format of specification.
- Parts of possible parameters:
  - the board array.

One space

Note format

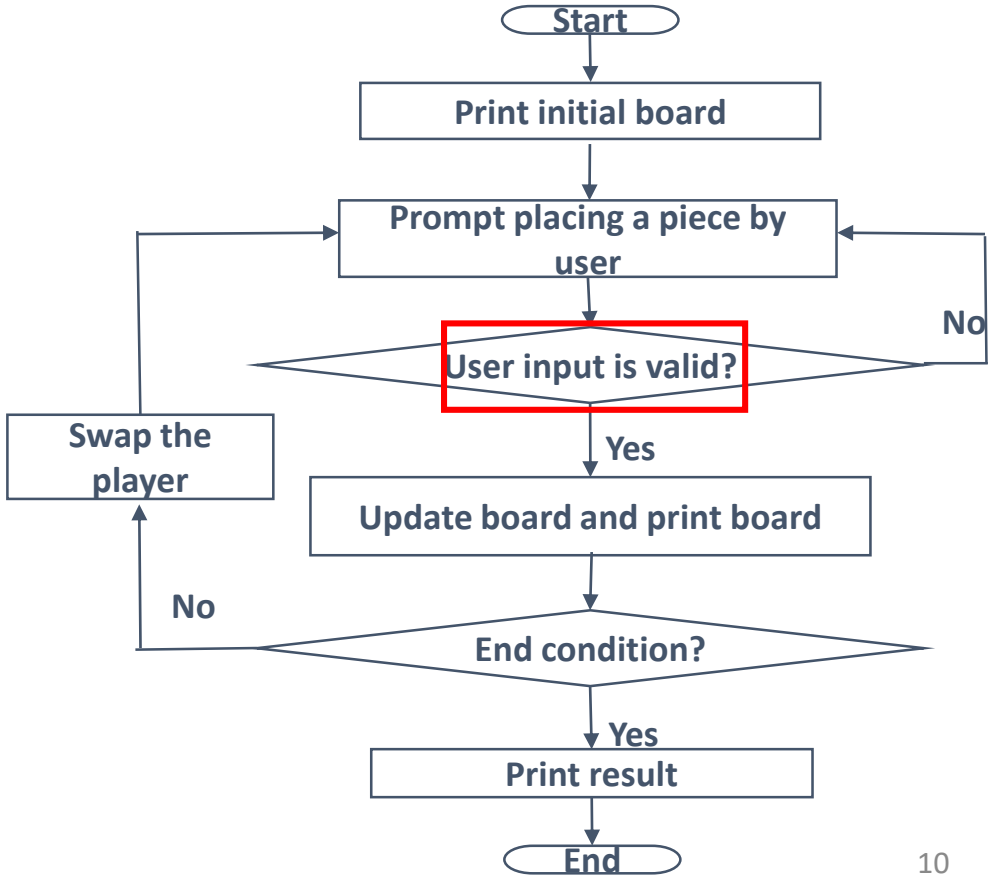
```
   A B C D E F G H I J K
 0 O X . X . . . . . . .
  1 . . . . . X . . . . .
   2 X . O . . . . . O X .
    3 . . . . . O . . . O X
     4 . X . . . . . . . . .
      5 O X X . O . . X . . O
       6 O . . . . . . . . X O
        7 . . . . . . . . . . .
         8 . O . . . . . X . . X
          9 . X . O . . . O . . .
          10 O . X . . . . . . O .
```

Note this alignment.
The ***setw()*** function may be a good choice.



Possible Functions

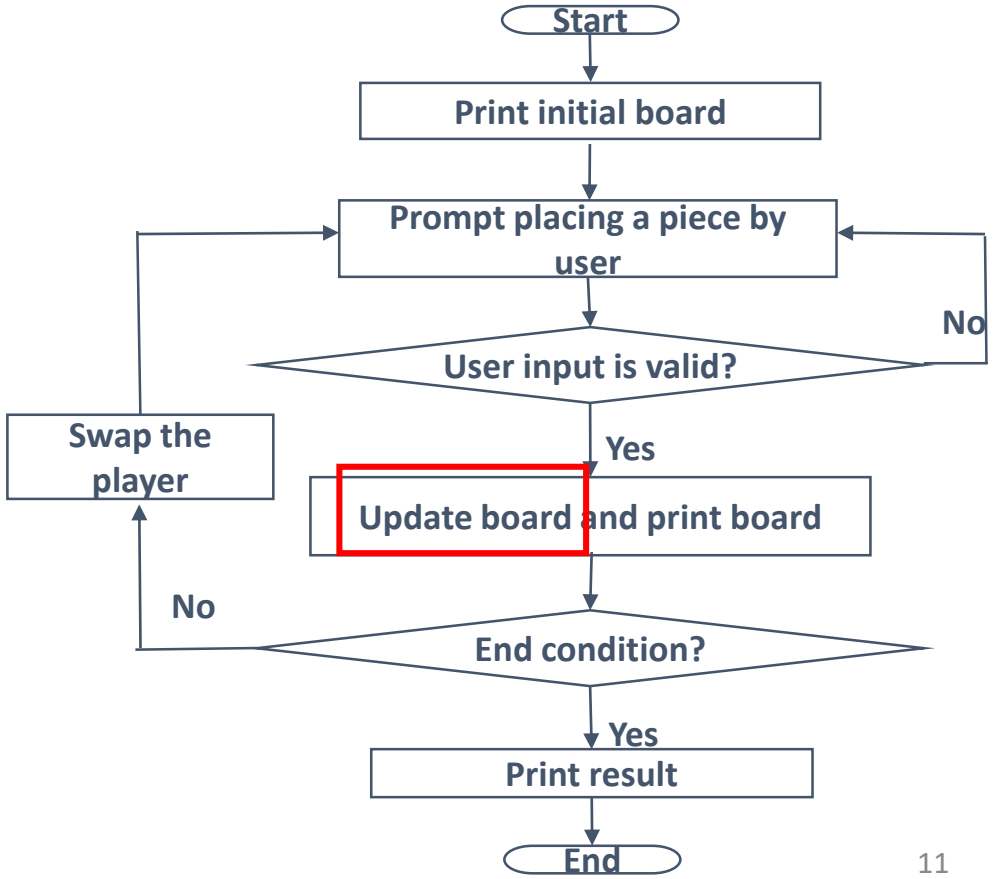
- A function checks whether the user input is valid. (You can assume that ***the inputs are always a character followed by an integer***)
- Parts of possible parameters:
  - the board array.
  - the user input
- A user input is invalid if:
  - (a) it is not a proper cell location (i.e., rows 0–10 and columns A–K, big letters only);
  - (b) the input location is already occupied.



Possible Functions

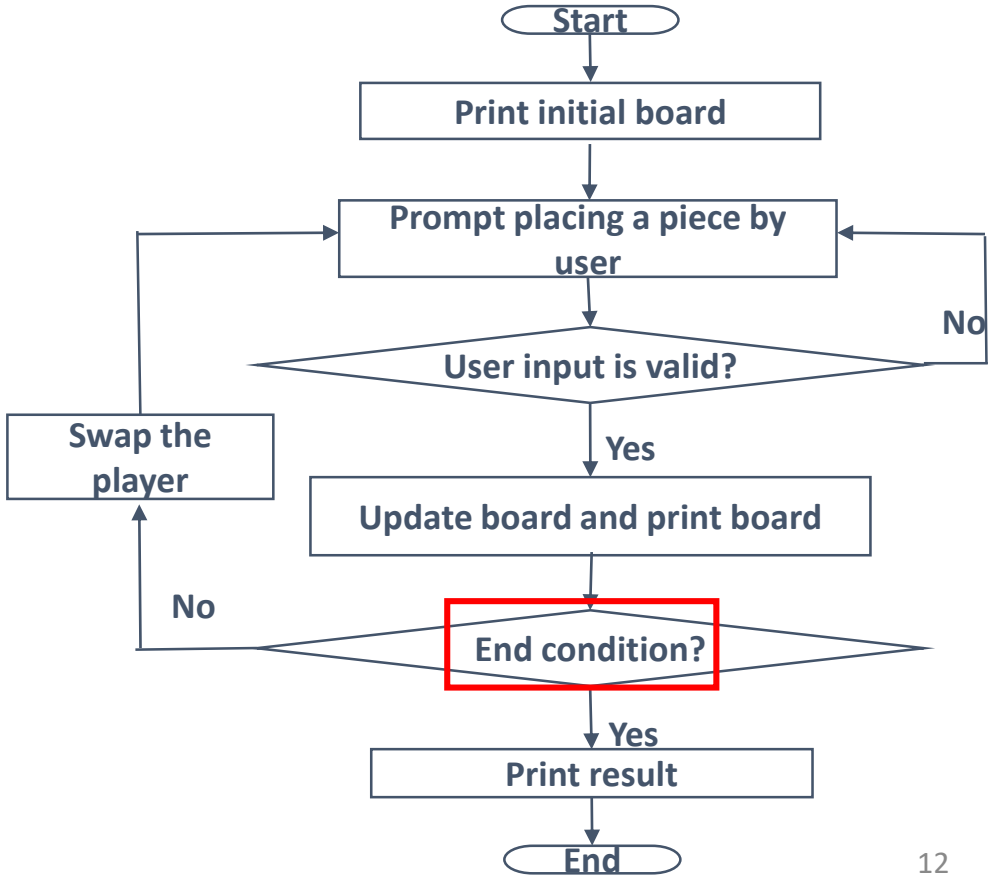
- A function updates the board.
- Parts of possible parameters:
  - the board array.
  - the position of the new piece
  - the current player

If the player is player O, the position needs to be placed an ‘O’.
If the player is player X, the position needs to be placed an ‘X’.



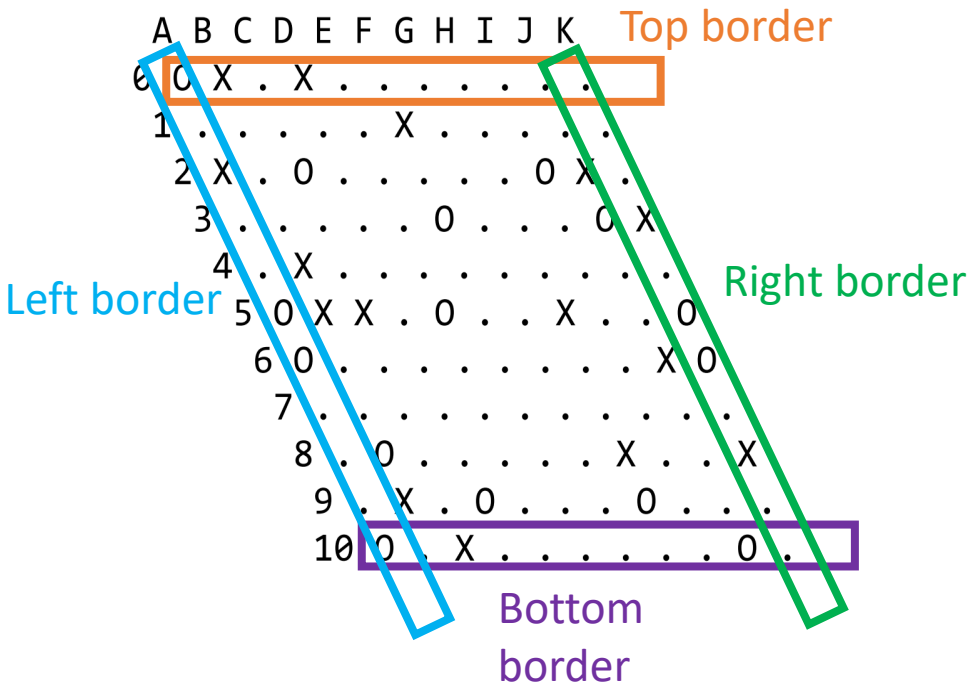
Possible Functions

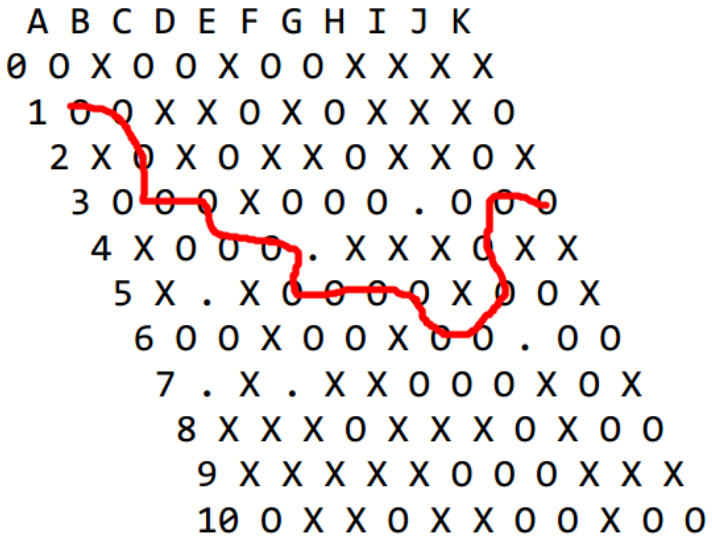
- A function checks whether the end condition appears. (Is a winning chain appears?)
- Parts of possible parameters:
  - the board array.
  - the position of the new piece
  - the current player



Check winning chain

- Winning condition definition:
  - Player O's winning chain: form a chain of his/her pieces connecting the **left** and **right** border of the board.
  - Player X's winning chain: form a chain of his/her pieces connecting the **top** and **bottom** border of the board.

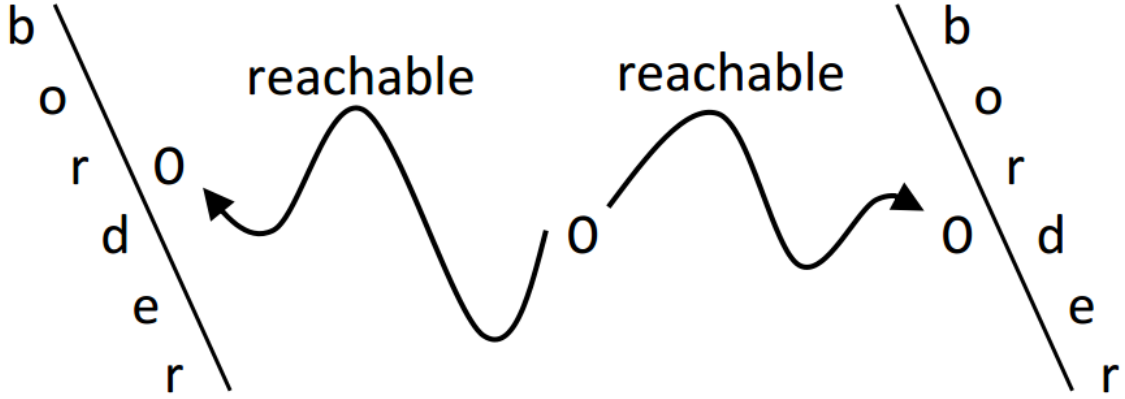




Example: the winning chain of player O (from left border to right border)

Check winning chain

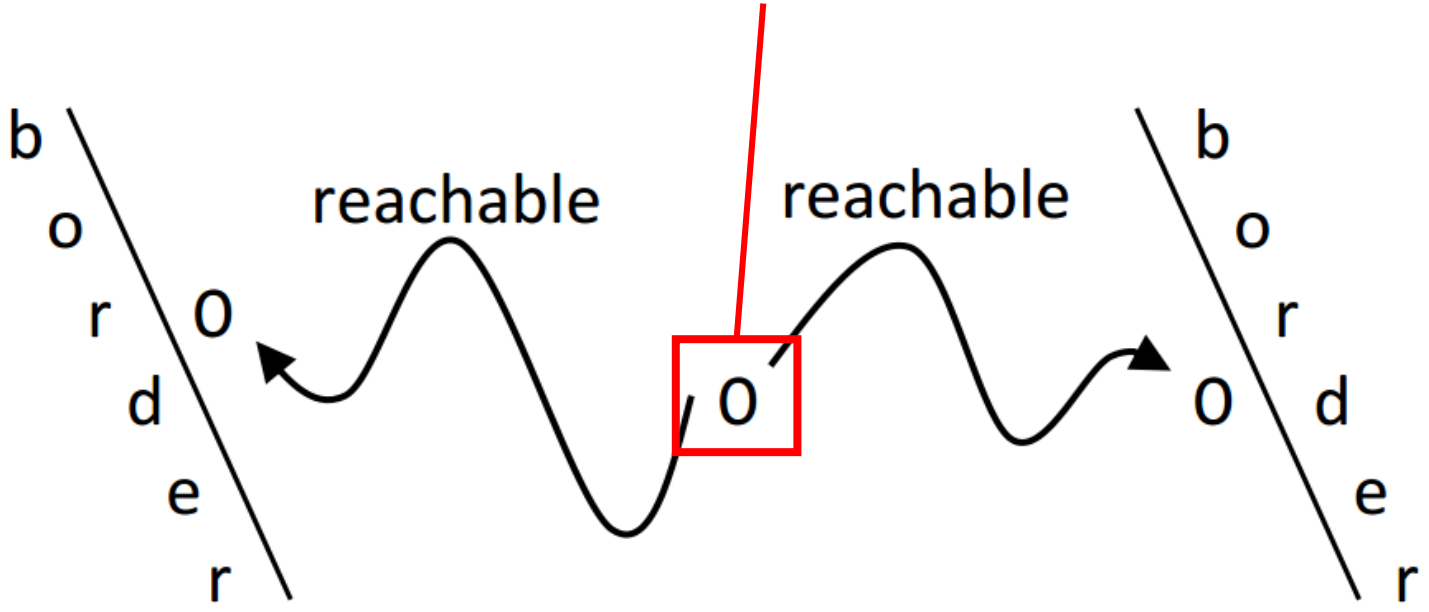
- Method:
  - Make a ***two-dimensional array*** *(reachable_array)* of type bool to store whether each cell location is "***reachable***" from the input location
  - Incrementally ***expand the reachability (set true) to neighbors*** of the input position, to neighbors of neighbors of the input position, to neighbors of neighbors of neighbors of the input position, and so on.
  - A player has formed a ***winning chain*** if you identify that the ***two corresponding borders*** are reached.



Check winning chain

- Method:

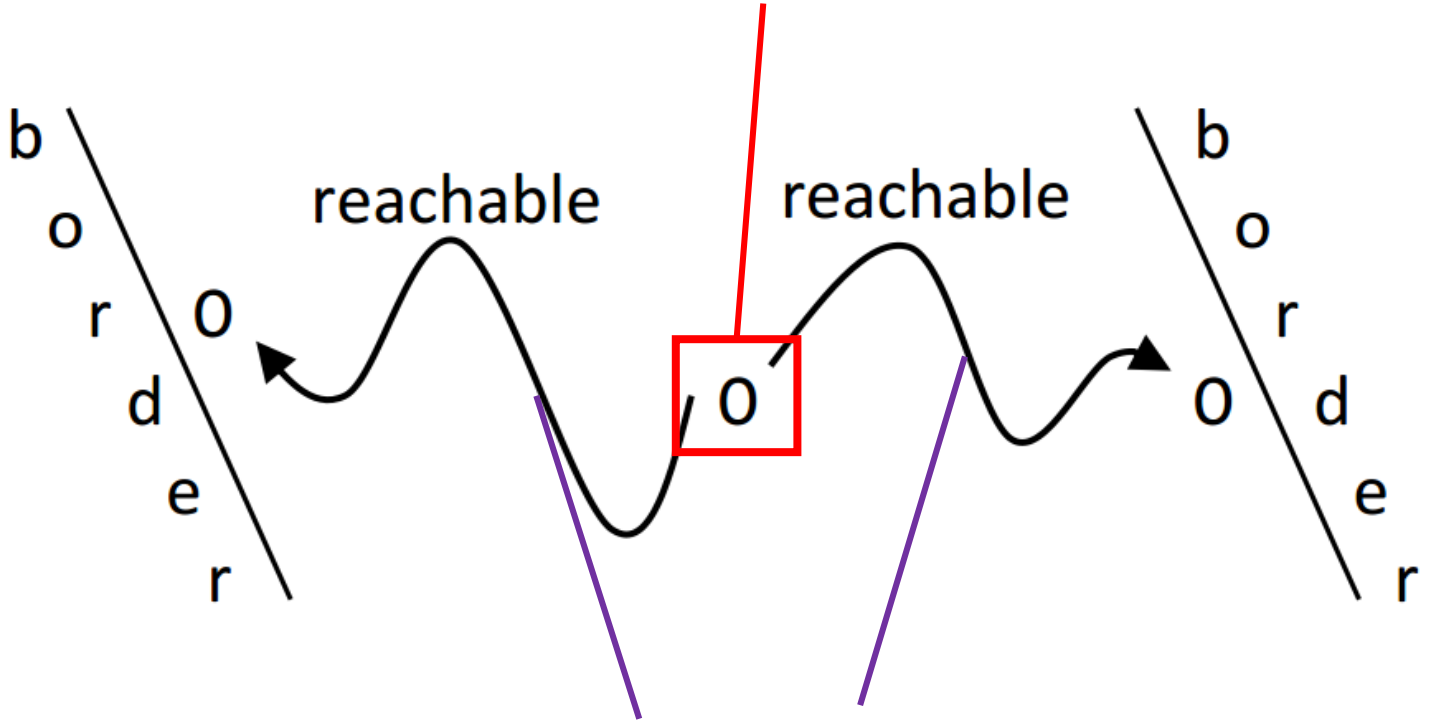
This is the input position.
This is also the first reachable cell (from the input position).



Check winning chain

- Method:

This is the input position.
This is also the first reachable cell (from the input position).

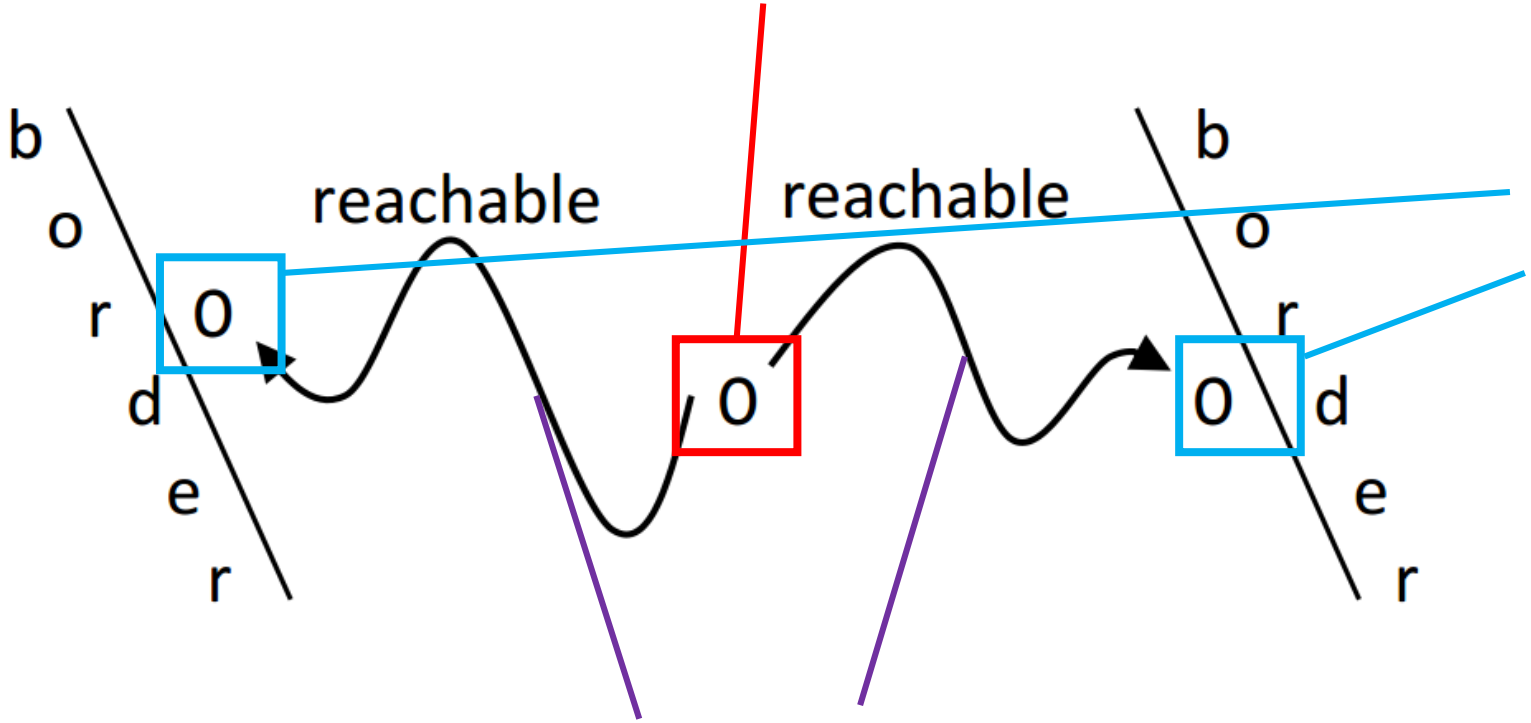


Expand the reachability to neighbors in all directions
(the reachability is recorded in the *reachable_array*)

Check winning chain

- Method:

This is the input position.
This is also the first reachable cell (from the input position).



Formed a winning chain if you identify that the two corresponding borders are reached

Expand the reachability to neighbors in all directions
(the reachability is recorded in the *reachable_array*)

Check winning chain

- Pseudo code

```
fun checkWinChain (board, input_position, other parameters ...) {

        initialize a 2-D reachable_array;

        for (each layer from the input_position)
                find reachable cells and setReachability to the 6 neighbors of these cells

        if (two corresponding borders are reached)
                return  this_palyer_wins;
        else
                return  game_continues;

}
```

```
 A B C D E F G H I J K
0 O X O O X O O X X X X
 1 O O X X O X O X X X O
  2 X O X O X X O X X O X
   3 O O O X O O O . O O O
    4 X O O O . X X X O X X
     5 X . X O O O O X O O X
      6 O O X O O X O O . O O
       7 . X . X X O O O X O X
        8 X X X O X X X O X O O
         9 X X X X X O O O X X X
         10 O X X O X X O O X O O
```

The input position and the first layer

Check winning chain

- Pseudo code

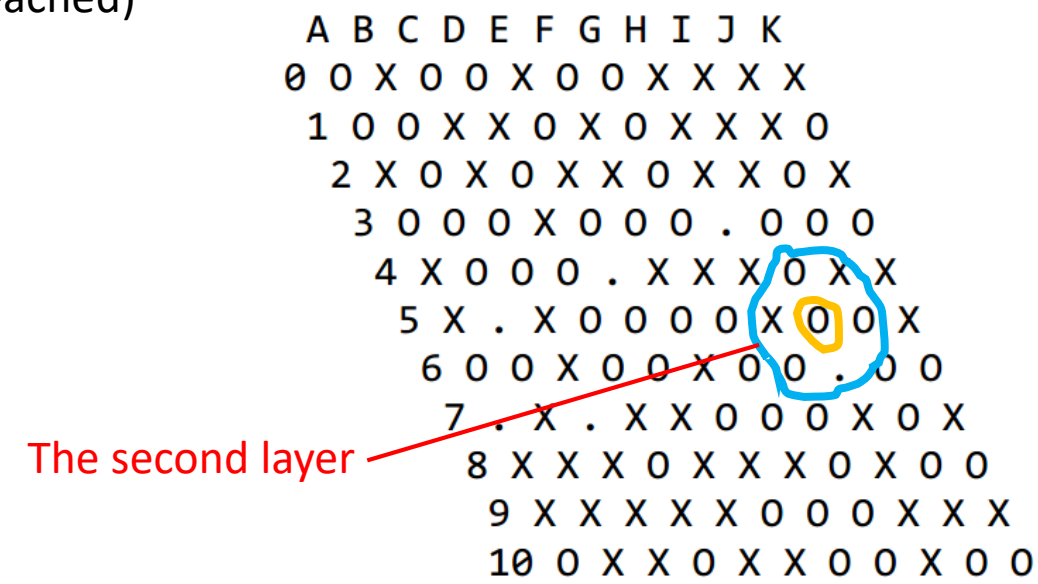
```
fun checkWinChain (board, input_position, other parameters ...) {

        initialize a 2-D reachable_array;

        for (each layer from the input_position)
                find reachable cells and setReachability to the 6 neighbors of these cells

        if (two corresponding borders are reached)
                return  this_palyer_wins;
        else
                return  game_continues;

}
```



Check winning chain

- Pseudo code

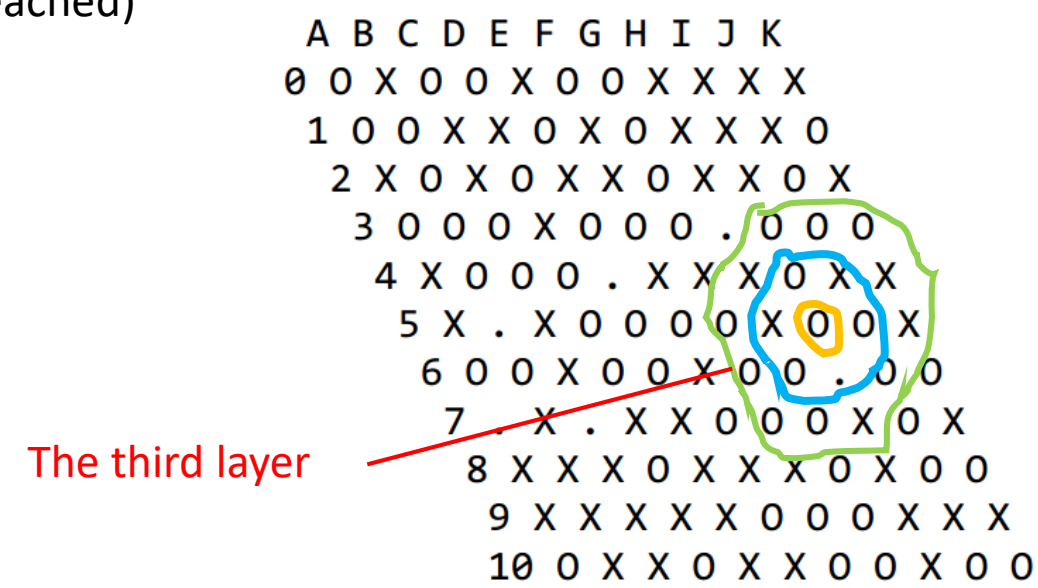
```
fun checkWinChain (board, input_position, other parameters ...) {

        initialize a 2-D reachable_array;

        for (each layer from the input_position)
                find reachable cells and setReachability to the 6 neighbors of these cells

        if (two corresponding borders are reached)
                return  this_palyer_wins;
        else
                return  game_continues;

}
```



Check winning chain

- Pseudo code

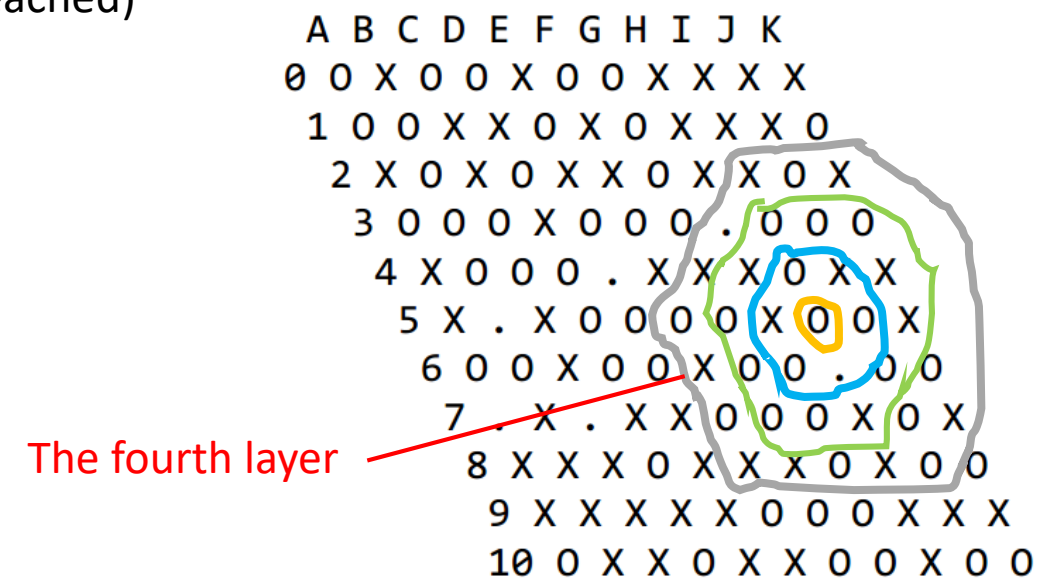
```
fun checkWinChain (board, input_position, other parameters ...) {

        initialize a 2-D reachable_array;

        for (each layer from the input_position)
                find reachable cells and setReachability to the 6 neighbors of these cells

        if (two corresponding borders are reached)
                return  this_palyer_wins;
        else
                return  game_continues;

}
```



Check winning chain

In the first layer, the reachability or the input_position is set to true; others are false.

```
 A B C D E F G H I J K
0 O X O O X O O X X X X
 1 O O X X O X O X X X O
  2 X O X O X X O X X O X
   3 O O O X O O O . O O O
    4 X O O O . X X X O X X
     5 X . X O O O O X O O X
      6 O O X O O X O O . O O
       7 . X . X X O O O X O X
        8 X X X O X X X O X O O
         9 X X X X X O O O X X X
         10 O X X O X X O O X O O
```

The input position and the first layer

	A	B	C	D	E	F	G	H	I	J	K
0	F	F	F	F	F	F	F	F	F	F	F
1	F	F	F	F	F	F	F	F	F	F	F
2	F	F	F	F	F	F	F	F	F	F	F
3	F	F	F	F	F	F	F	F	F	F	F
4	F	F	F	F	F	F	F	F	F	F	F
5	F	F	F	F	F	F	F	F	T	F	F
6	F	F	F	F	F	F	F	F	F	F	F
7	F	F	F	F	F	F	F	F	F	F	F
8	F	F	F	F	F	F	F	F	F	F	F
9	F	F	F	F	F	F	F	F	F	F	F
10	F	F	F	F	F	F	F	F	F	F	F

reachable_array

Check winning chain

```
 A B C D E F G H I J K
0 O X O O X O O X X X X
 1 O O X X O X O X X X O
  2 X O X O X X O X X O X
   3 O O O X O O O . O O O
    4 X O O O . X X X O X X
     5 X . X O O O O X O O X
      6 O O X O O X O O . O O
       7 . X . X X O O O X O X
        8 X X X O X X X O X O O
         9 X X X X X O O O X X X
         10 O X X O X X O O X O O
```

The second layer

These cells have been set to true already.

	A	B	C	D	E	F	G	H	I	J	K
0	F	F	F	F	F	F	F	F	F	F	F
1	F	F	F	F	F	F	F	F	F	F	F
2	F	F	F	F	F	F	F	F	F	F	F
3	F	F	F	F	F	F	F	F	F	F	F
4	F	F	F	F	F	F	F	F	T	F	F
5	F	F	F	F	F	F	F	F	T	T	F
6	F	F	F	F	F	F	F	T	F	F	F
7	F	F	F	F	F	F	F	F	F	F	F
8	F	F	F	F	F	F	F	F	F	F	F
9	F	F	F	F	F	F	F	F	F	F	F
10	F	F	F	F	F	F	F	F	F	F	F

reachable_array

These cells is set to true in this layer.

Check winning chain

```
 A B C D E F G H I J K
0 O X O O X O O X X X X
 1 O O X X O X O X X X O
  2 X O X O X X O X X O X
   3 O O O X O O O . O O O
    4 X O O O . X X X O X X
     5 X . X O O O O X O O X
      6 O O X O O X O O . O O
       7 . X . X X O O O X O X
        8 X X X O X X X O X O O
         9 X X X X X O O O X X X
         10 O X X O X X O O X O O
```

The third layer

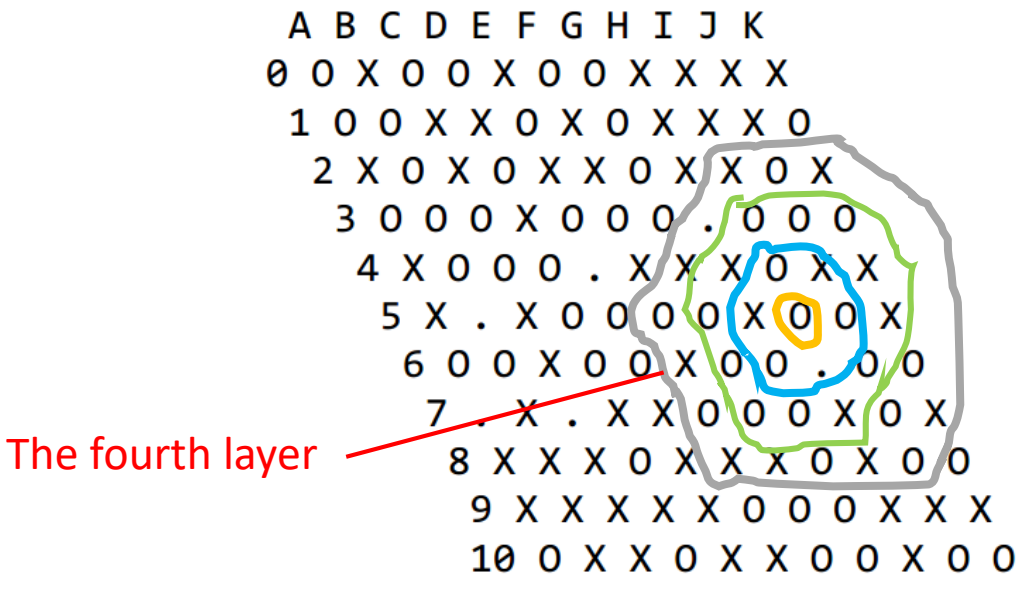
These cells have been set to true already.

	A	B	C	D	E	F	G	H	I	J	K
0	F	F	F	F	F	F	F	F	F	F	F
1	F	F	F	F	F	F	F	F	F	F	F
2	F	F	F	F	F	F	F	F	F	F	F
3	F	F	F	F	F	F	F	F	T	T	T
4	F	F	F	F	F	F	F	F	T	F	F
5	F	F	F	F	F	F	T	F	T	T	F
6	F	F	F	F	F	F	T	T	F	T	F
7	F	F	F	F	F	F	T	T	F	F	F
8	F	F	F	F	F	F	F	F	F	F	F
9	F	F	F	F	F	F	F	F	F	F	F
10	F	F	F	F	F	F	F	F	F	F	F

reachable_array

These cells is set to true in this layer.

Check winning chain



These cells have been set to true already.

	A	B	C	D	E	F	G	H	I	J	K
0	F	F	F	F	F	F	F	F	F	F	F
1	F	F	F	F	F	F	F	F	F	F	F
2	F	F	F	F	F	F	F	F	F	T	F
3	F	F	F	F	F	F	F	F	T	T	T
4	F	F	F	F	F	F	F	F	T	F	F
5	F	F	F	F	F	T	T	F	T	T	F
6	F	F	F	F	F	F	T	T	F	T	T
7	F	F	F	F	F	T	T	T	F	T	F
8	F	F	F	F	F	F	F	T	F	T	F
9	F	F	F	F	F	F	F	F	F	F	F
10	F	F	F	F	F	F	F	F	F	F	F

reachable_array

These cells is set to true in this layer.

Check winning chain

```
 A B C D E F G H I J K
0 O X O O X O O X X X X
 1 O O X X O X O X X X O
  2 X O X O X X O X X O X
   3 O O O X O O O . O O O
    4 X O O O . X X X O X X
     5 X . X O O O O X O O X
      6 O O X O O X O O . O O
       7 . X . X X O O O X O X
        8 X X X O X X X O X O O
         9 X X X X X O O O X X X
         10 O X X O X X O O X O O
```

The fourth layer

These cells have been set to true already.

	A	B	C	D	E	F	G	H	I	J	K
0	F	F	F	F	F	F	F	F	F	F	F
1	F	F	F	F	F	F	F	F	F	F	F
2	F	F	F	F	F	F	F	F	F	T	F
3	F	F	F	F	F	F	F	F	T	T	T
4	F	F	F	F	F	F	F	F	T	F	F
5	F	F	F	F	F	T	T	F	T	T	F
6	F	F	F	F	F	F	T	T	F	T	T
7	F	F	F	F	F	T	T	T	F	T	F
8	F	F	F	F	F	F	F	T	F	T	F
9	F	F	F	F	F	F	F	F	F	F	F
10	F	F	F	F	F	F	F	F	F	F	F

reachable_array

These cells is set to true in this layer.

Check winning chain

```
 A B C D E F G H I J K
0 O X O O X O O X X X X
 1 O O X X O X O X X X O
  2 X O X O X X O X X O X
   3 O O O X O O O . O O O
    4 X O O O . X X X O X X
     5 X . X O O O O X O O X
      6 O O X O O X O O . O O
       7 . X . X X O O O X O X
        8 X X X O X X X O X O O
         9 X X X X X O O O X X X
         10 O X X O X X O O X O O
```

Finally:

	A	B	C	D	E	F	G	H	I	J	K
0	F	F	F	F	F	F	F	F	F	F	F
1	F	F	F	F	T	F	T	F	F	F	T
2	F	T	F	T	F	F	T	F	F	T	F
3	T	T	T	F	T	T	T	F	T	T	T
4	F	T	T	T	F	F	F	F	T	F	F
5	F	F	F	T	T	T	T	F	T	T	F
6	F	F	F	T	T	F	T	T	F	T	T
7	F	F	F	F	F	T	T	T	F	T	F
8	F	F	F	F	F	F	F	T	F	T	T
9	F	F	F	F	F	T	T	T	F	F	F
10	F	F	F	F	F	F	T	T	F	F	F

reachable_array

Check winning chain

- Pseudo code

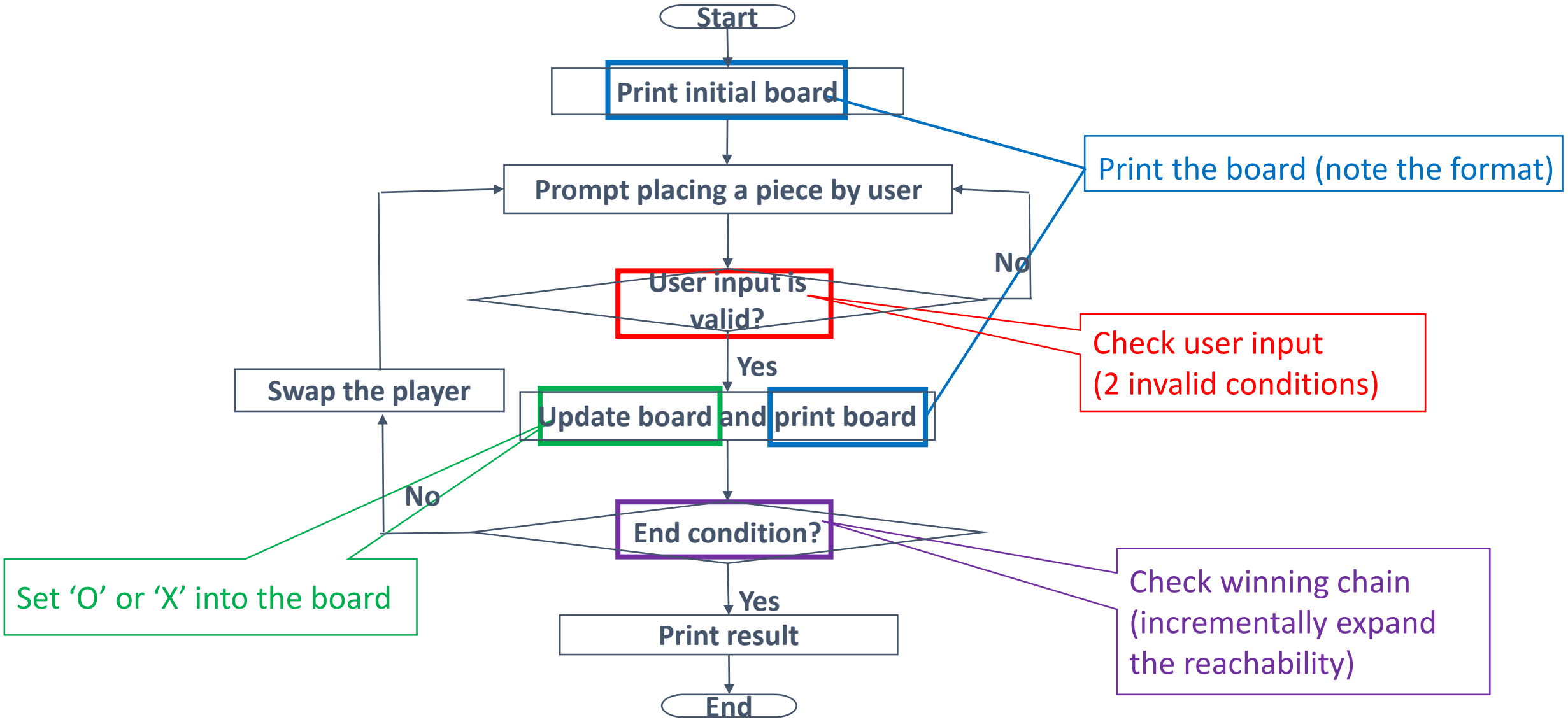
```
fun setReachability (a cell position, other parameters …) {
        if (this cell is invalid)
                give up and go back;
        if (this cell has been reached)  // by checking the reachable_array
                give up and go back;
        if (this cell cannot be reached)   // for example: this cell is occupied by the opposite
                give up and go back;
        // other end condition if needed

        mark this cell as reachable;  // mark in the reachable_array

        if (this cell is a border of this player)
                record it;  // through reference parameters or returned values

}
```

Summary


Start
Print initial board
Prompt placing a piece by user
User input is valid?
No
Yes
Swap the player
Update board and print board
No
End condition?
Yes
Print result
End
Print the board (note the format)
Check user input
(2 invalid conditions)
Set 'O' or 'X' into the board
Check winning chain
(incrementally expand
the reachability)

Thank You!

Q&A