CSCI 1540 Introduction to Computing Using C++

Tutorial 08

Xiaopeng Zhang

SHB 913

xpzhang@cse.cuhk.edu.hk

Outline

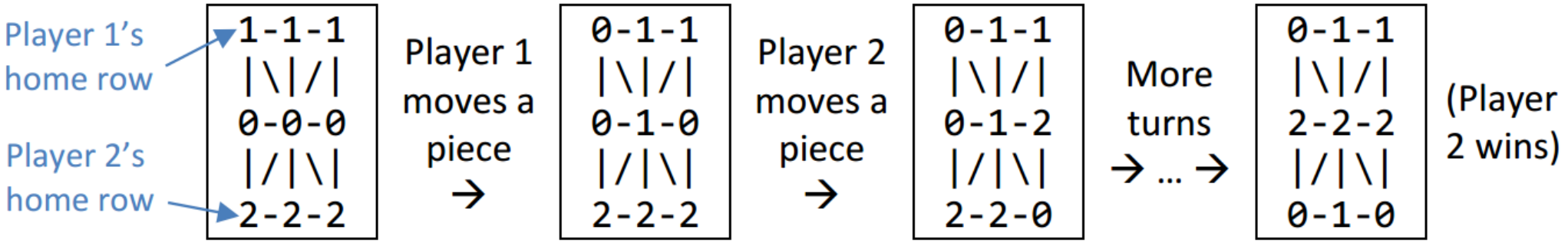
- Assignment 4: Tant Fant

Requirement

- Assignment 4: Tant Fant
- Deadline: 20:00, Thu 7 Nov 2019
- Requirements:
  - Filename: tantfant.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;
  - No global variables (variables declared outside any functions);
  - No functions in the <cmath> library;
  - No arrays.

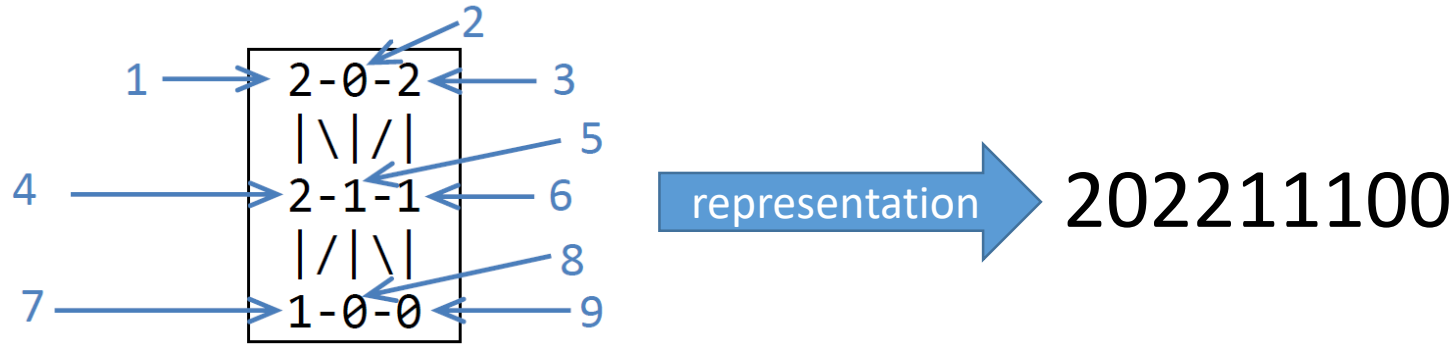
Game Description

- Start condition:
  - There are 2 players. Each player has three pieces at their own home rows.
- Game stage:
  - The players take turn to move one of their pieces to an adjacent empty position. *(Player 1 takes the first turn.)*
- End condition:
  - The player who first aligns their three pieces **horizontally**, **vertically**, or **diagonally** (except his/her home row) wins.



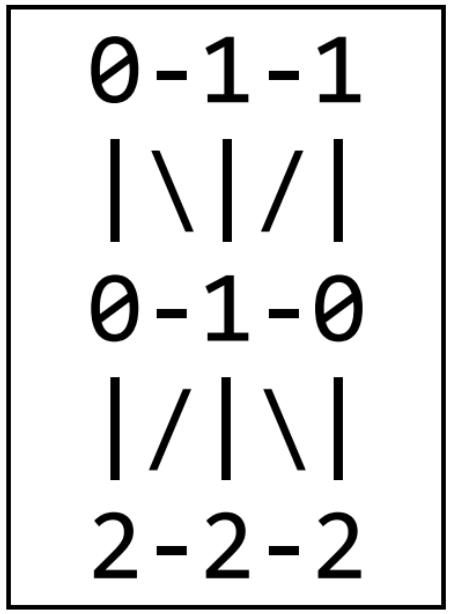
Board Representation

- There are 9 possible positions in a board. We use a 9-digit integer $d_1d_2d_3d_4d_5d_6d_7d_8d_9$ **(do not use array)** to denote these board positions.
  - Each digit d_i is either 0, 1, or 2
  - Values 1 or 2 mean position i is a piece of players 1 or 2 respectively. Value 0 means position i is empty.
  - The initial board is encoded as 111000222.
  - in C++, integer constants should NOT contain leading zeroes. Example: not 011222010, but 11222010.



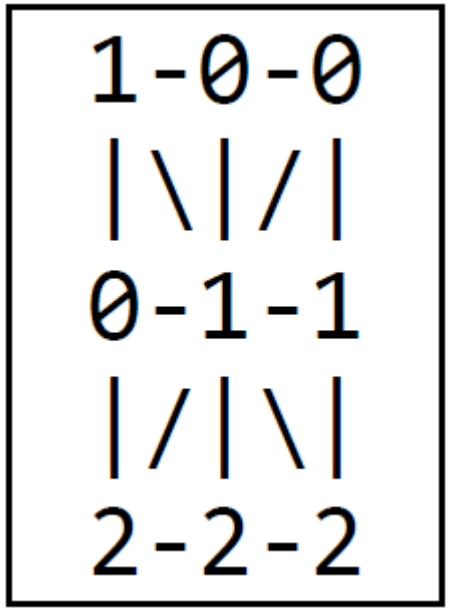
Board Representation

- Examples for board representation:



~~011010222~~

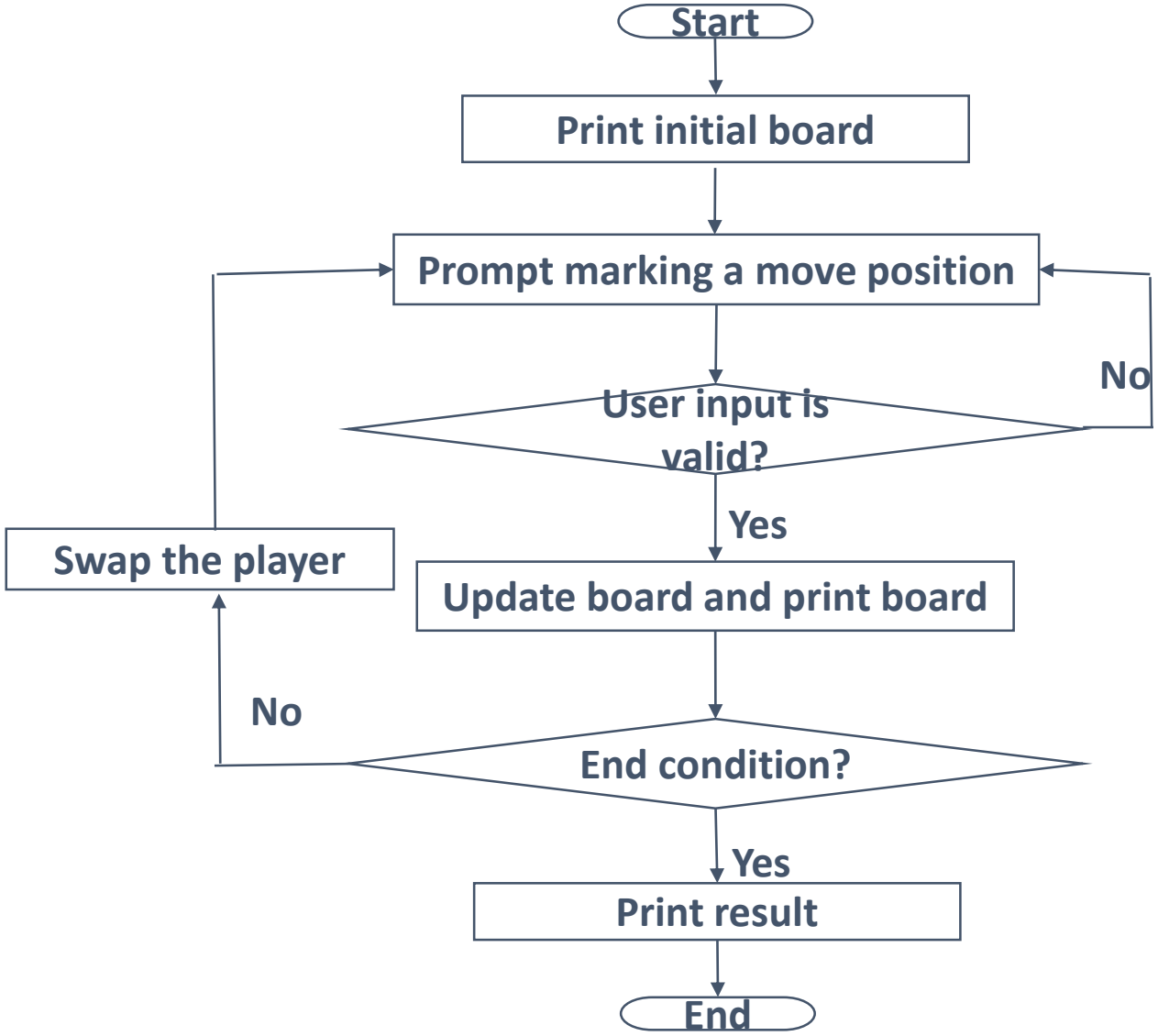
11010222



100011222

In C++, integer constants should NOT contain leading zeroes; otherwise, they will be treated as octal numbers.

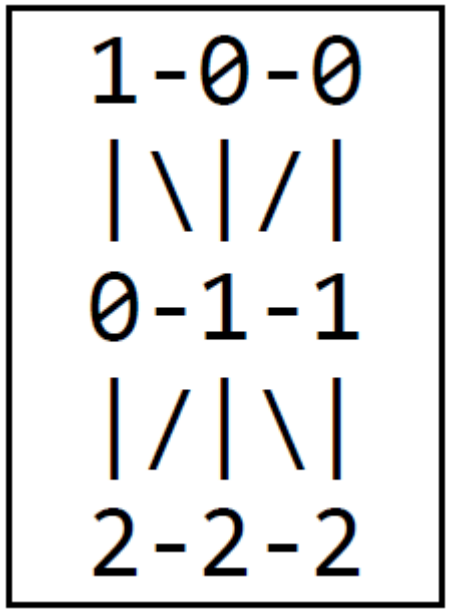
Program Flow



Useful Given Functions

- ***(Provided)*** int *status* (int board, int pos)
- Returns 0 when the position is empty, 1 when the position is a piece of player 1, and 2 when the position is a piece of player 2.

```
int status(int board, int pos) {
    for (int i = 0; i < 9 - pos; i++)
    board /= 10;
    return board % 10;
}
```



board = 100011222
status(*board*, 6) == 1
status(*board*, 5) == 1
status(*board*, 7) == 2
status(*board*, 2) == 0

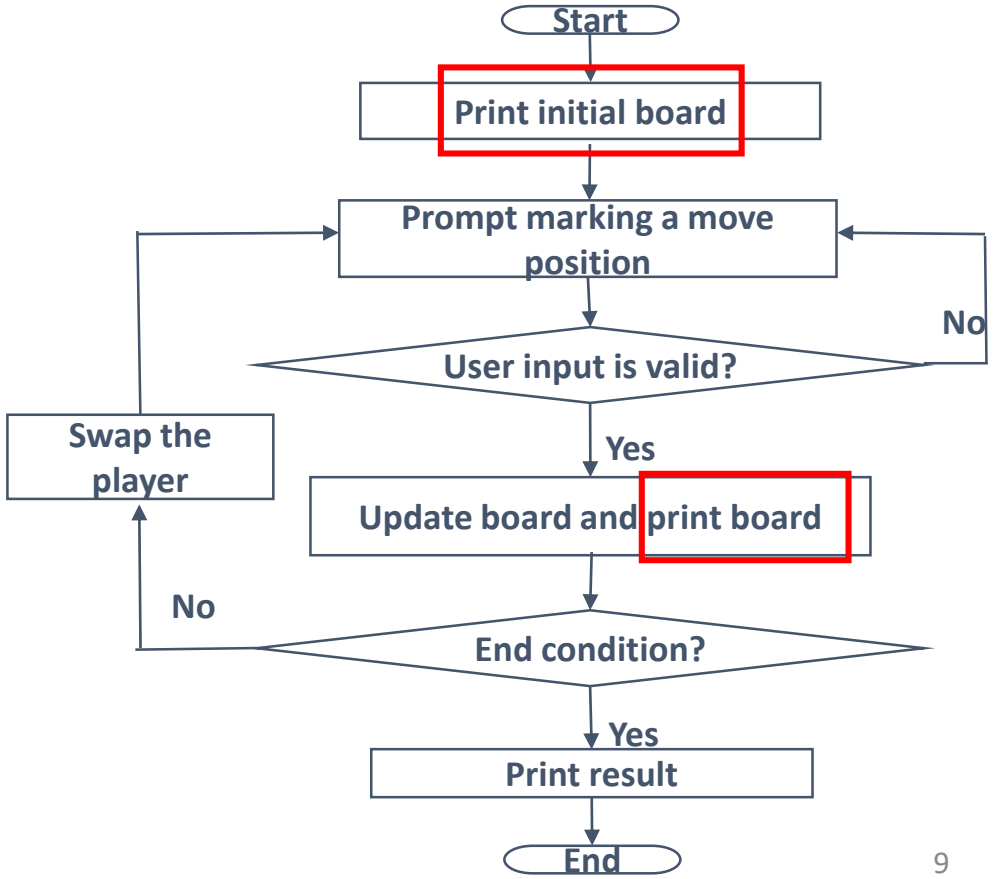
This function may be called in many procedures in the program flow.

Functions that might be used

- ***(Provided)*** void *printBoard* (int board).
- This function prints the board to the screen using the format of specification.

printBoard(200211000);

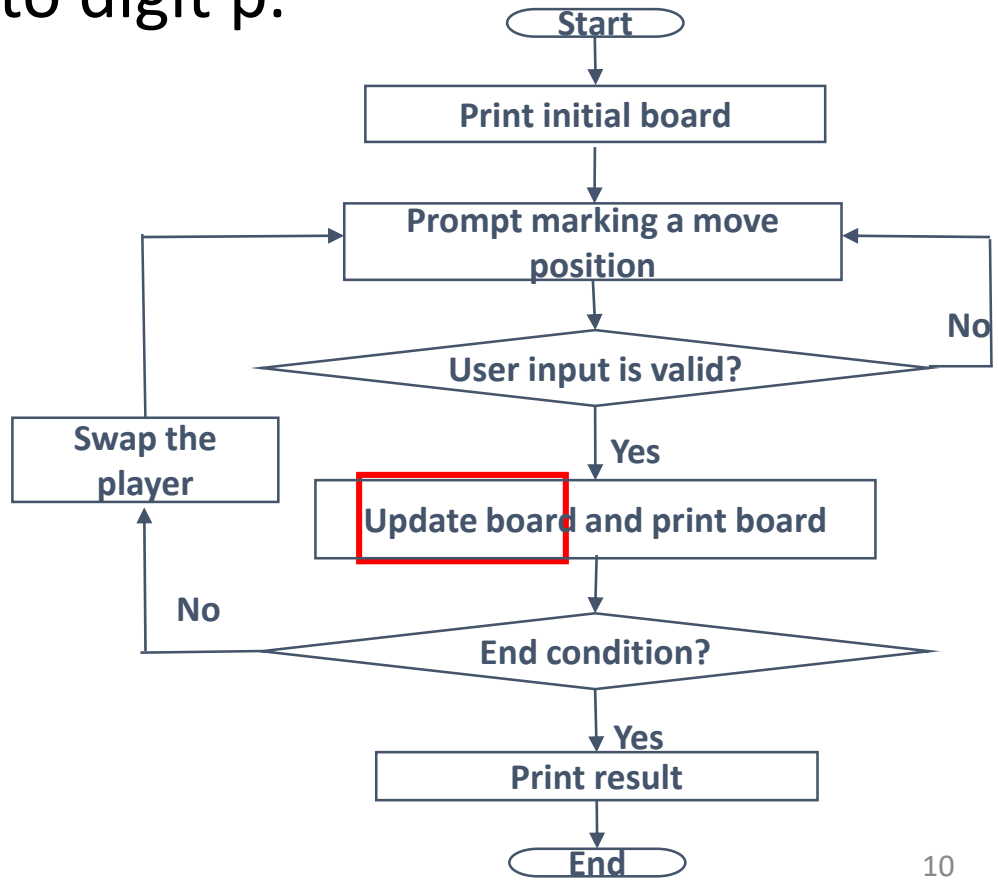
```
2-0-0
|\|/|
2-1-1
|/|\|
0-0-0
```



Required Functions

- void *updateBoard* (int &board, int from, int to, int p) – Required!
- This function updates the from^{th} digit of the reference parameter board to 0, and the to^{th} digit of board to digit p.

At the end of this function, the value of `board` becomes the new board configuration *after* the piece movement.



Required Functions

- void *updateBoard* (int &board, int from, int to, int p) – Required!
- This function updates the from^{th} digit of the reference parameter board to 0, and the to^{th} digit of board to digit p.

board	from	to	p	Expected return value of movePiece(board, from, to, p)
202211100	6	2	1	212210100
100211220	4	9	2	100011222
2011122	8	4	2	2211102

Note that you do *not* have to check in this function whether the piece movement is valid or not. You just have to update the digits in the two relevant positions.

Required Functions

- void *updateBoard* (int &board, int from, int to, int p) – Required!

board	from	to	p	Expected return value of movePiece(board, from, to, p)
202211100	6	2	1	212210100
100211220	4	9	2	100011222
2011122	8	4	2	2211102

Example:
from = 4, to = 9, p = 2

Make the to-th digit to p:

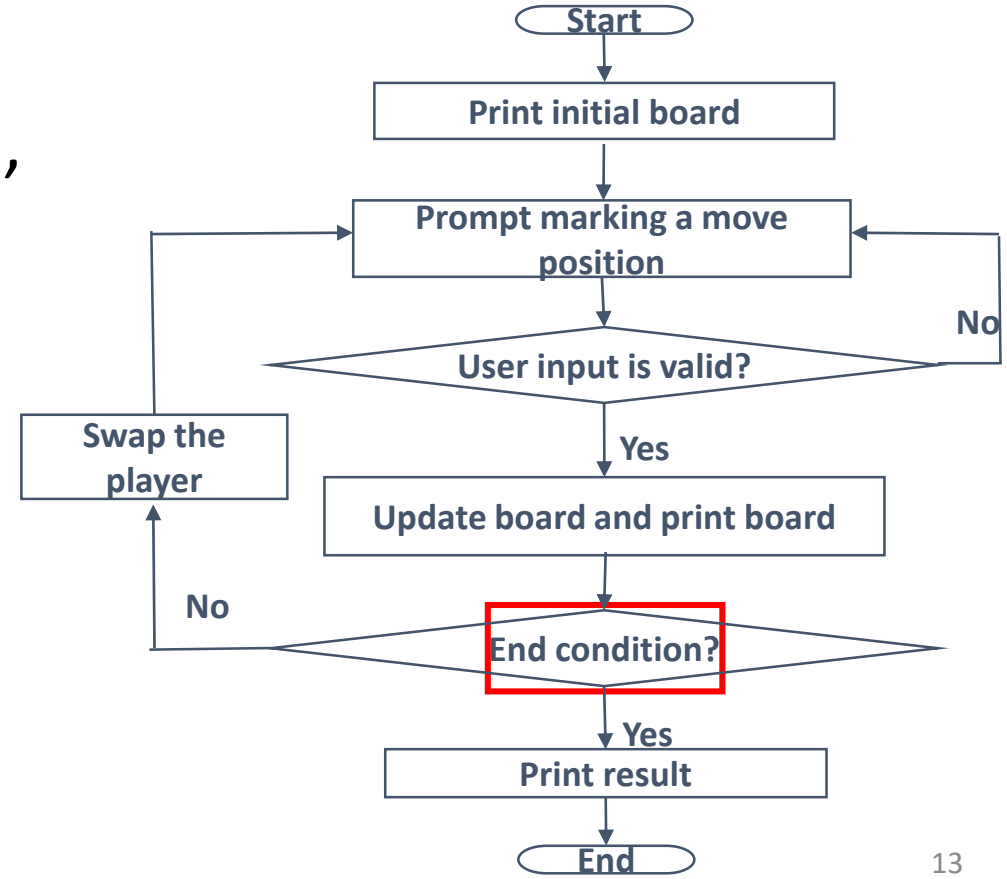
```
int factor = 1;
for (int i = 0; i < 9 - 9; i++)
        factor *= 10;
board += 2 * factor;
```

Make the from-th digit to 0

```
int factor = 1;
for (int i = 0; i < 9 - 4; i++)
        factor *= 10;
board -= 2 * factor;
```

Required Functions

- int *formLine* (int board) – Required!
- This function checks if any player has formed a line **horizontally**, **vertically**, or **diagonally** in board *except his/her home row*.
  If so, the function returns either 1 or 2, meaning the player of the line.
  Otherwise, the function returns 0, meaning no line is formed.

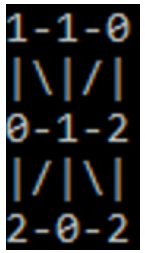


Required Functions

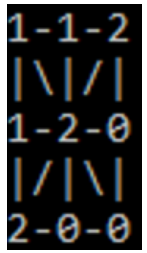
- int *formLine* (int board) – Required!
- This function checks if any player has formed a line **horizontally**, **vertically**, or **diagonally** in board *except his/her home row*. If so, the function returns either 1 or 2, meaning the player of the line. Otherwise, the function returns 0, meaning no line is formed.

Examples:

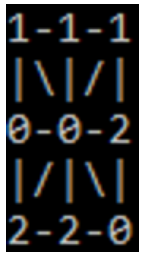
(1) formLine (110012202) should return 0
(2) formLine (112120200) should return 2
(3) formLine (111002220) should return 0 -> *his/her home row*
(4) formLine (202200111) should return 1 -> *the opponent's home row*



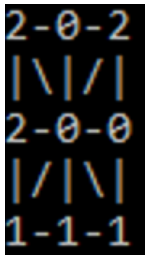
(1)



(2)



(3)



(4)

Required Functions

- int *formLine* (int board) – Required
- Content:

(1) Check horizontally: pos -> (1,2,3), (4,5,6), (7,8,9). Any row all equals to 1 (or all equals to 2), then return player 1 (or 2) wins *except his/her home row*

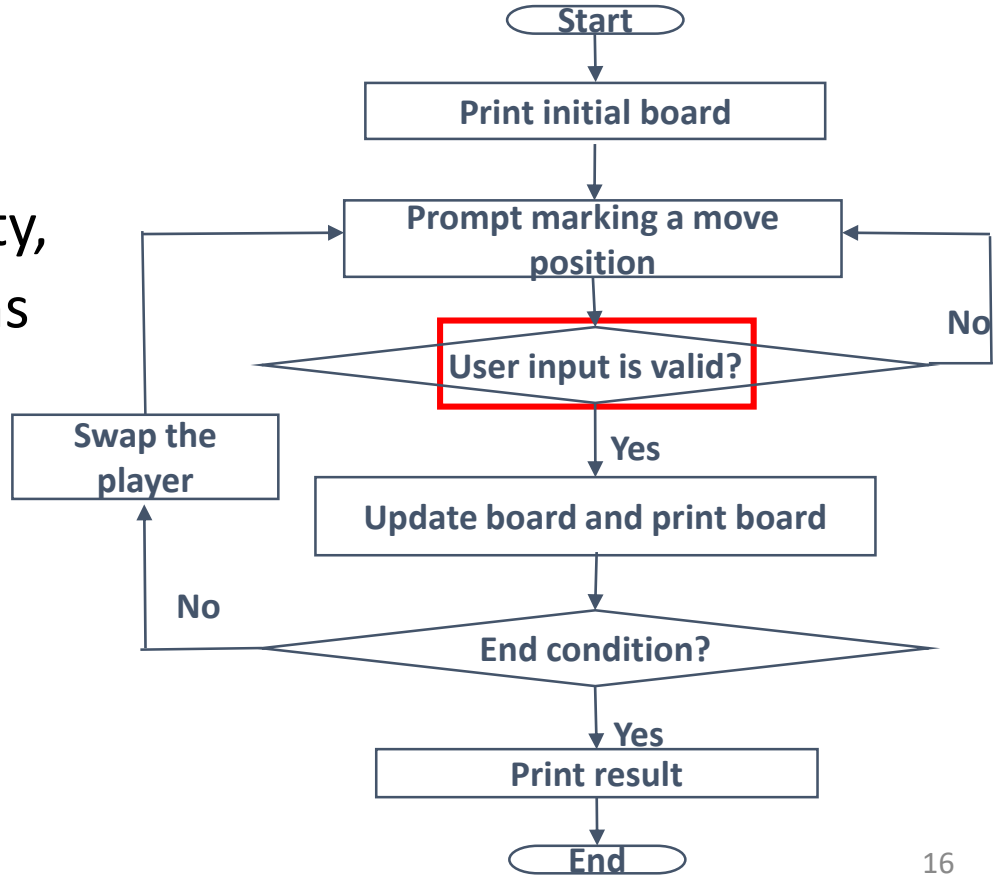
(2) Check vertically: pos -> (1,4,7), (2,5,8), (3,6,9). Any column all equals to 1 (or all equals to 2), then return player 1 (or 2) wins

(3) Check diagonally: pos -> (1, 5, 9), (3, 5, 7). Any diagonal line all equals to 1 (or all equals to 2), then return player 1 (or 2) wins

(4) If do not form a line, return 0

Check valid input

- Check if the user input is valid.
- A user input is invalid if
  - (a) two input positions are not 1–9,
  - (b) or the source position is not occupied by the current player,
  - (c) or the destination position is not empty,
  - (d) or the source and destination positions are not adjacent.



Check valid input

- Check if the user input is valid.
- A user input is invalid if
  - (a) two input positions are not 1–9,
  - (b) or the source position is not occupied by the current player,
  - (c) or the destination position is not empty,
  - (d) or the source and destination positions are not adjacent.

Examples:

```
1-1-0
|\|/|
0-1-0
|/|\|
2-2-2
```

(current board)

```
Player 2 (from to): 0 4↵
Invalid. Try again!
```

Source position is outside 1–9

Check valid input

- Check if the user input is valid.
- A user input is invalid if
  - (a) two input positions are not 1–9,
  - (b) or the source position is not occupied by the current player,
  - (c) or the destination position is not empty,
  - (d) or the source and destination positions are not adjacent.

Examples:

```
1-1-0
|\|/|
0-1-0
|/|\|
2-2-2
```

(current board)

```
Player 2 (from to): 2 3↵
Invalid. Try again!
```

Position 2 is not a piece of player 2

Check valid input

- Check if the user input is valid.
- A user input is invalid if
  - (a) two input positions are not 1–9,
  - (b) or the source position is not occupied by the current player,
  - (c) or the destination position is not empty,
  - (d) or the source and destination positions are not adjacent.

Examples:

```
1-1-0
|\|/|
0-1-0
|/|\|
2-2-2
```

(current board)

```
Player 2 (from to): 6 3↵
Invalid. Try again!
```

Position 6 is empty; not a piece of player 2

Check valid input

- Check if the user input is valid.
- A user input is invalid if
  - (a) two input positions are not 1–9,
  - (b) or the source position is not occupied by the current player,
  - (c) or the destination position is not empty,
  - (d) or the source and destination positions are not adjacent.

Examples:

```
1-1-0
|\|/|
0-1-0
|/|\|
2-2-2
```

(current board)

Position 5 has been occupied by player 1

```
Player 2 (from to): 7 5↵
Invalid. Try again!
```

Check valid input

- Check if the user input is valid.
- A user input is invalid if
  - (a) two input positions are not 1–9,
  - (b) or the source position is not occupied by the current player,
  - (c) or the destination position is not empty,
  - (d) or the source and destination positions are not adjacent.

Examples:

```
1-1-0
|\|/|
0-1-0
|/|\|
2-2-2
```

(current board)

Source and destination are not adjacent

```
Player 2 (from to): 8 6↵
Invalid. Try again!
```

Check valid input

- Check if the user input is valid.
- A user input is invalid if
  - (a) two input positions are not 1–9,
  - (b) or the source position is not occupied by the current player,
  - (c) or the destination position is not empty,
  - (d) or the source and destination positions are not adjacent.

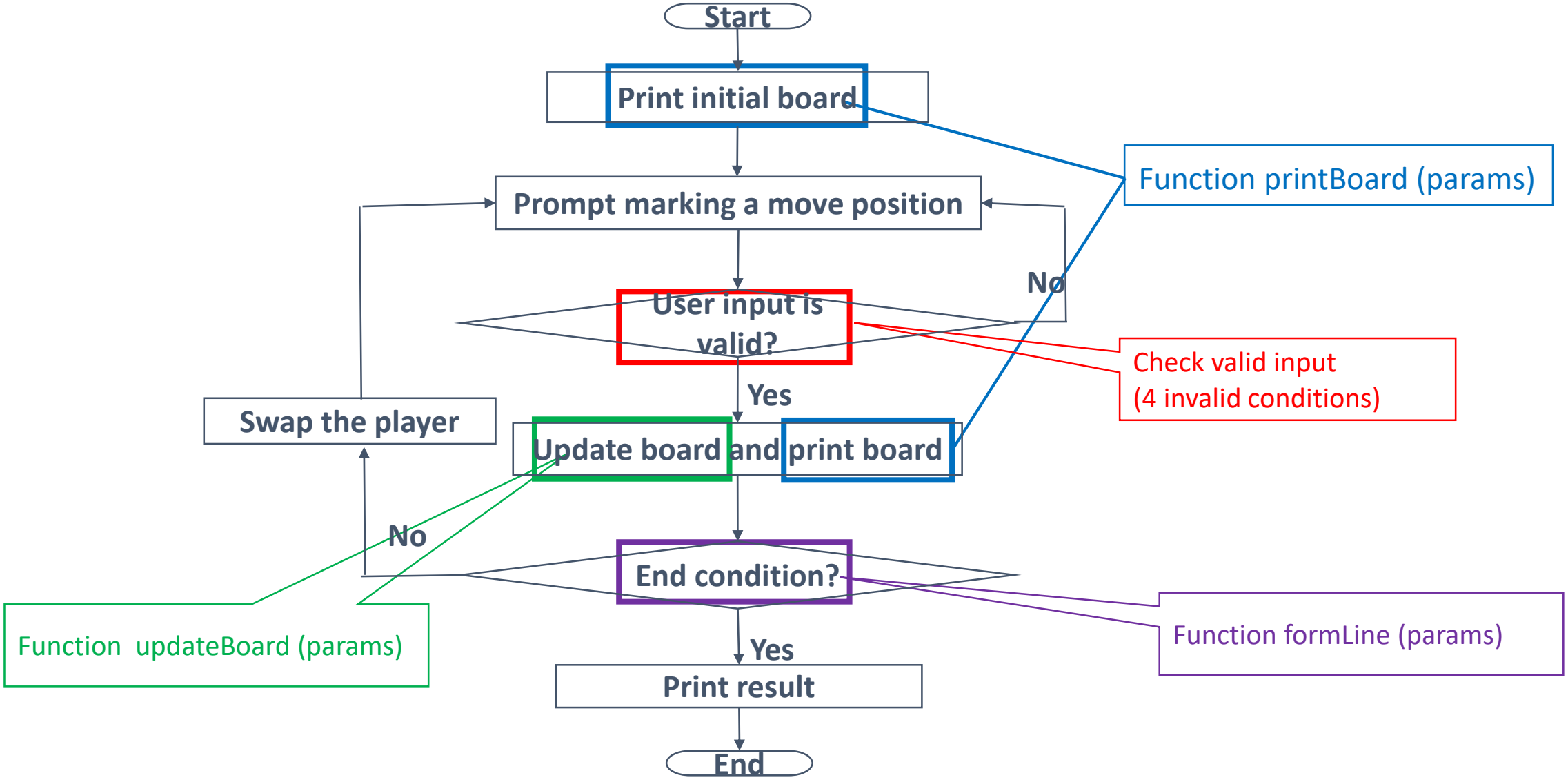
Examples:

```
1-1-0
|\|/|
0-1-0
|/|\|
2-2-2
```

If the from-th digit is **1**, the to-th digit can only be **2,4,5**.
If the from-th digit is **2**, the to-th digit can only be **1,3,5**.
If the from-th digit is **3**, the to-th digit can only be **2,5,6**.
……

```
if ((from == 1 && (to == 2 || to == 4 || to == 5)) ||
    (from == 2 && (to == 3 || to == 5)) ||
    …… )
```

Summary


Start
Print initial board
Prompt marking a move position
User input is valid?
No
Yes
Swap the player
Update board and print board
No
End condition?
Yes
Print result
End
Function printBoard (params)
Check valid input
(4 invalid conditions)
Function updateBoard (params)
Function formLine (params)

Notes

- Do NOT modify the prototypes of the required and provided functions.
- Besides required and provided functions, you can add other function by yourself if necessary.

Thank You!

Q&A