ESKİŞEHİR OSMANGAZİ ÜNİVERSİTESİ

MÜHENDİSLİK MİMARLIK FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

INTRODUCTION TO PROGRAMMING LAB.
DÖNEM PROJESİ

Öğretim Üyesi:

Dr. Öğr. Üyesi İbrahim SAVRAN

Hazırlayan:

Suude Kaynak - 152120211110

Ocak 2024

The Description of the Problem

3 Ships are randomly placed on a 10x10 table. Two players are asked to make a move respectively. If the player's shot hits all three of the opponent's ships, the player wins the game. Also The player is warned ıf he wants to shoot at an area where a move has been made before . This game can be played on different size of tables, not just 10x10.

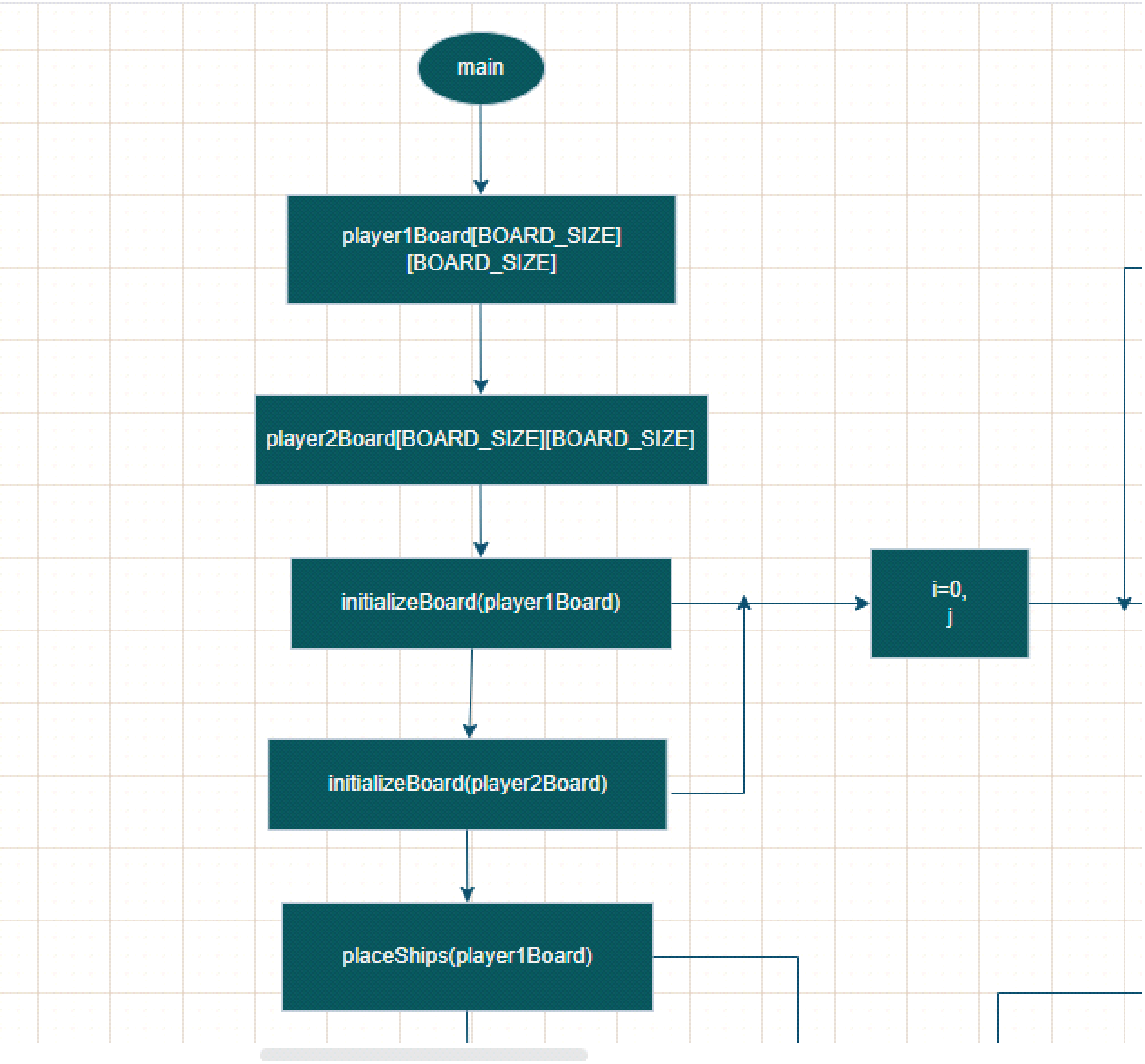
The Method Applied While Solving the Problem

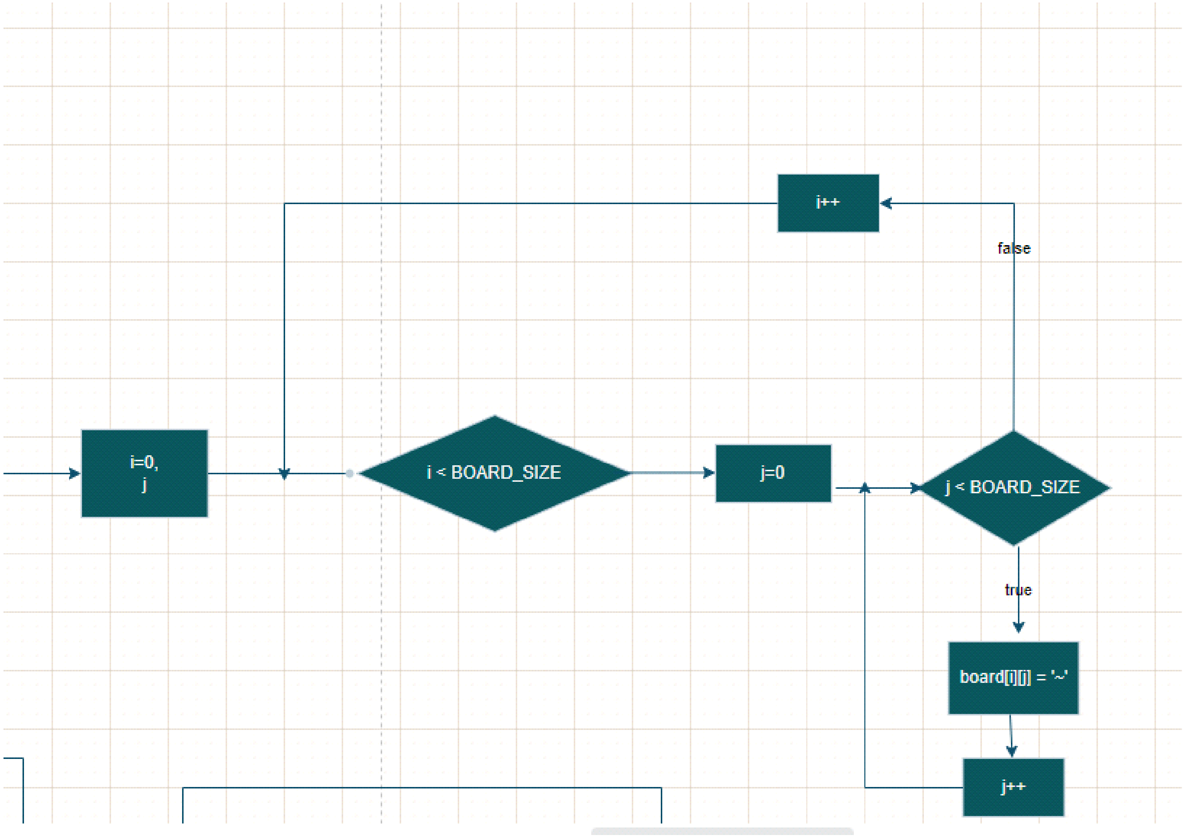
–

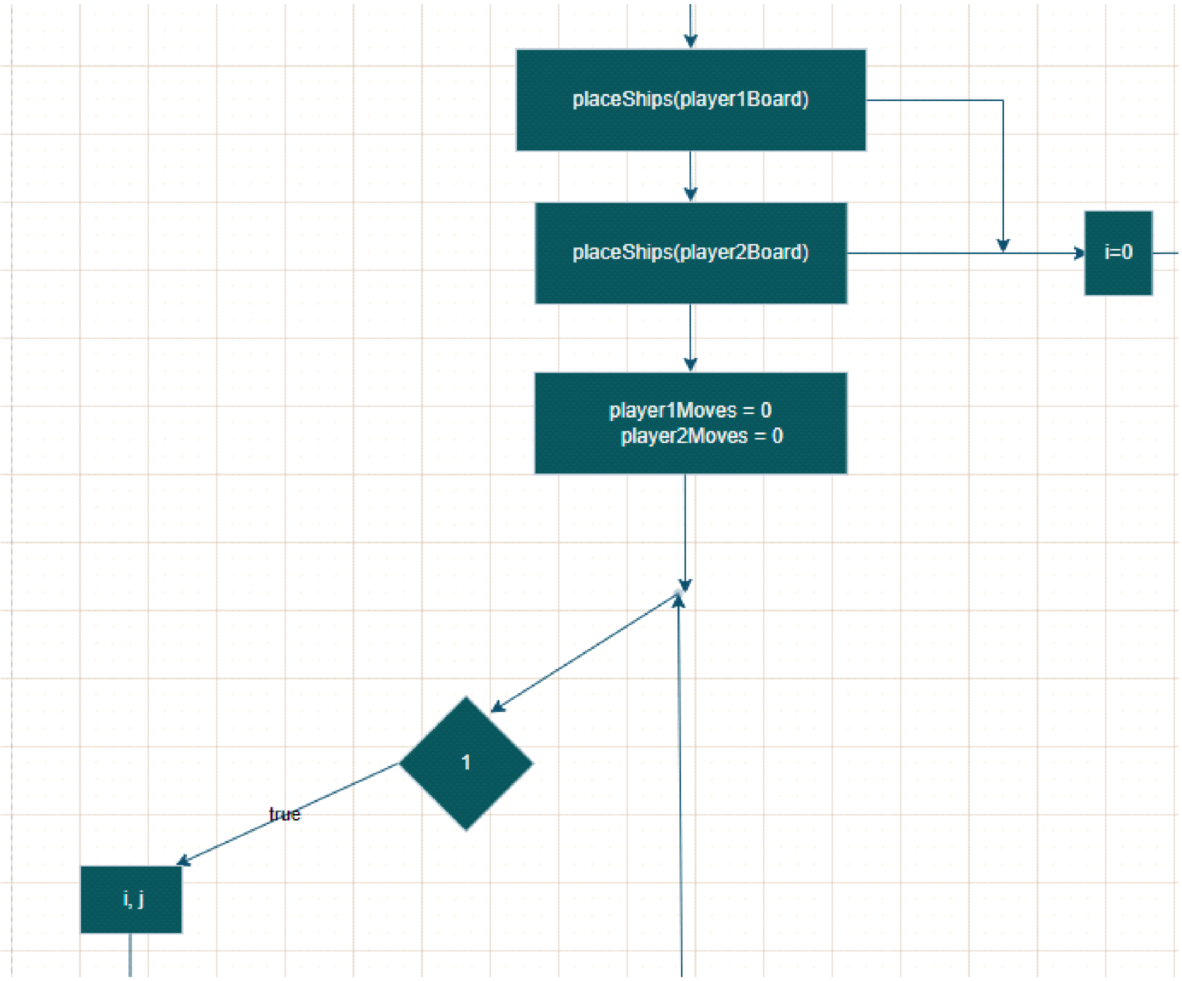
First of all, I defined the size of the table as 10,   so the game can be played on different sizes of table, not just 10. Then I created a function that fill the table with the '~' symbol for each coordinate.   I created a function that randomly places 3 ships on different places on the table. I created two functions to print the table I created before. One of them prints the table with ships. Firstly I called this function and showed the locations of their ships to players. Then I called the other print function each time the players make a move, this function doesn't show the locations of the ships to opponent player. I also defined two more functions that check wheter the entered move is valid or inavlid and the ships is hit or not. I called these funtion where needed in the main function. I asked the players to make a move respectively. If the players hit all three of the opponent's ships, I displayed the name of the winner on the screen.

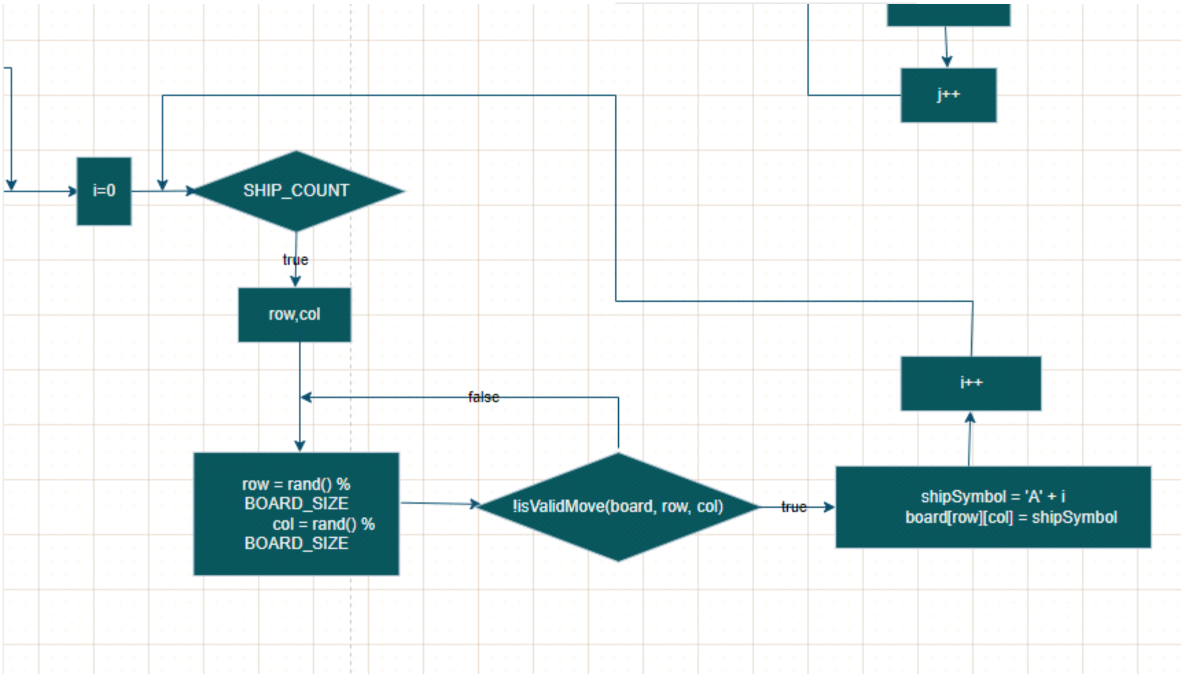
The Flowchart of the Game

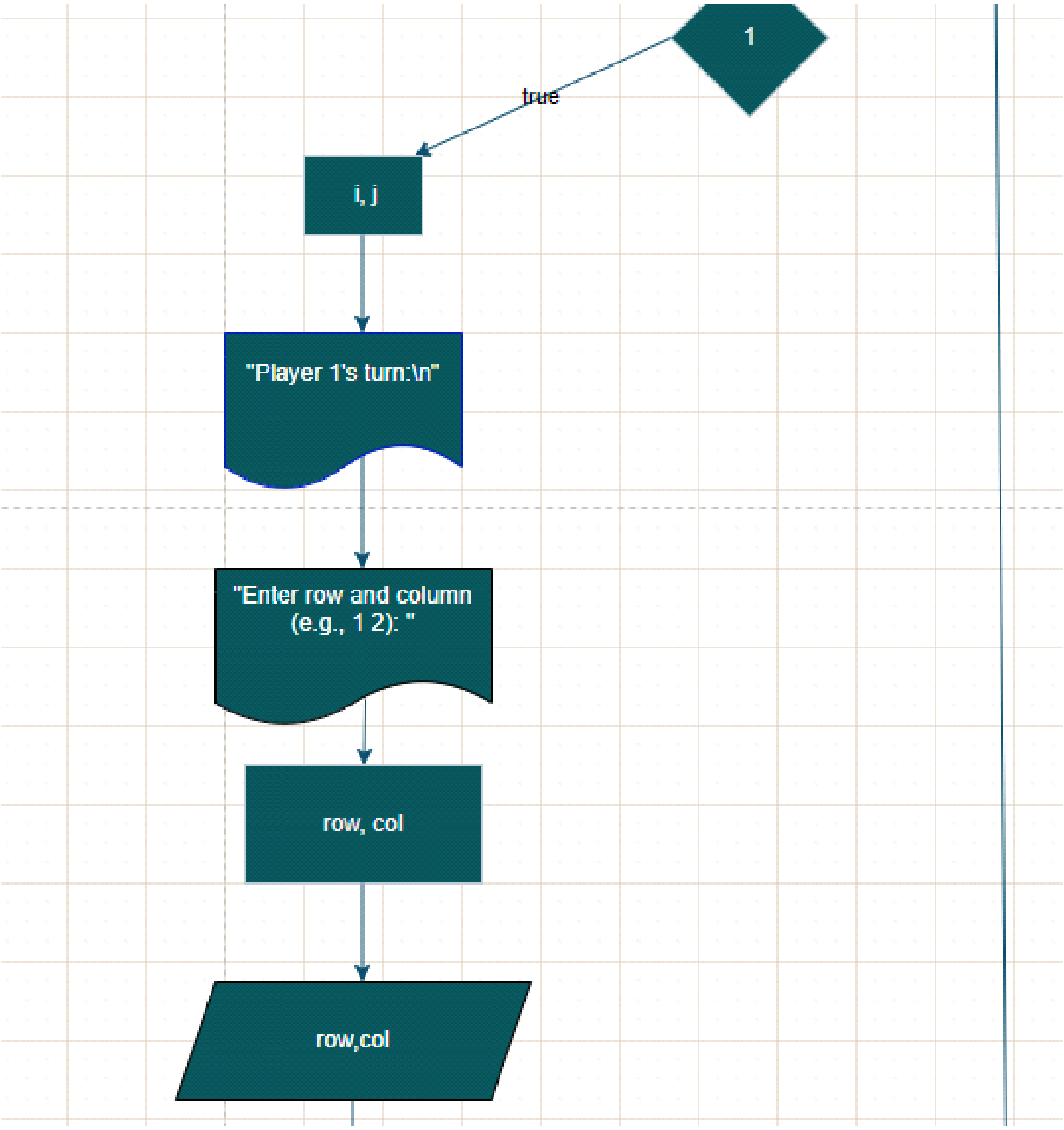
(I also added the draw.io file of this flowchart in the folder.)

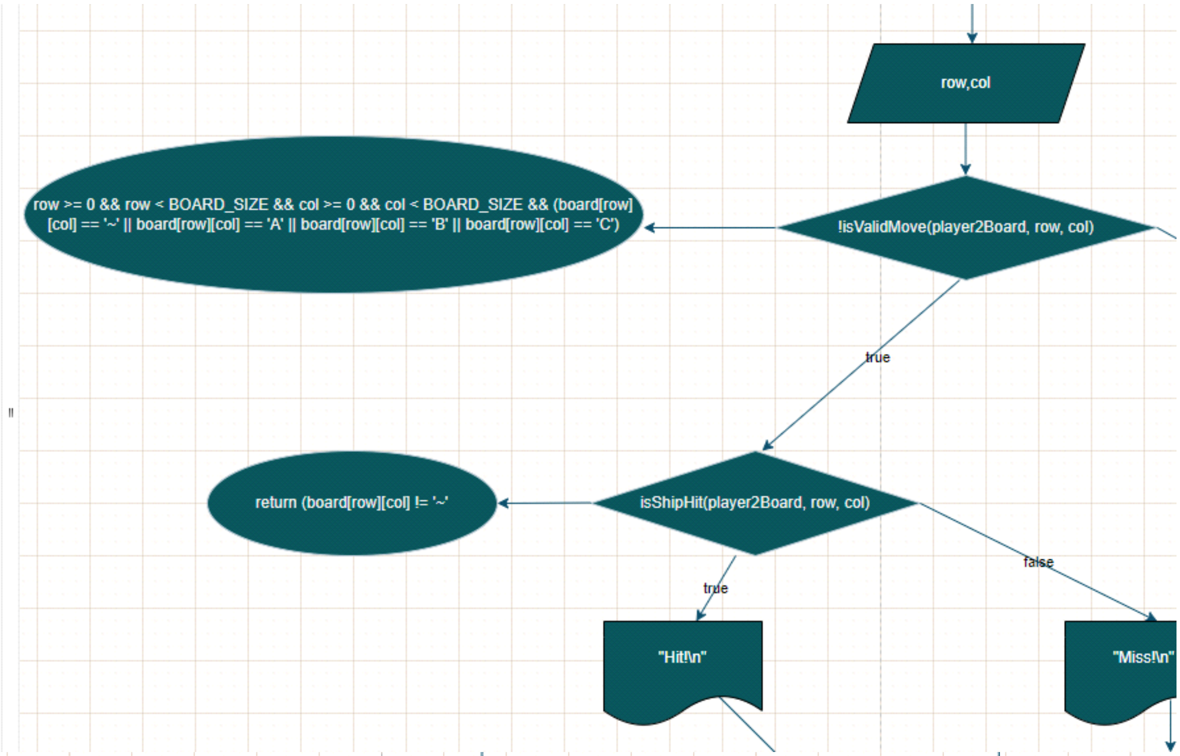

main
player1Board[BOARD_SIZE]
[BOARD_SIZE]
player2Board[BOARD_SIZE][BOARD_SIZE]
initializeBoard(player1Board)
i=0,
j
initializeBoard(player2Board)
placeShips(player1Board)

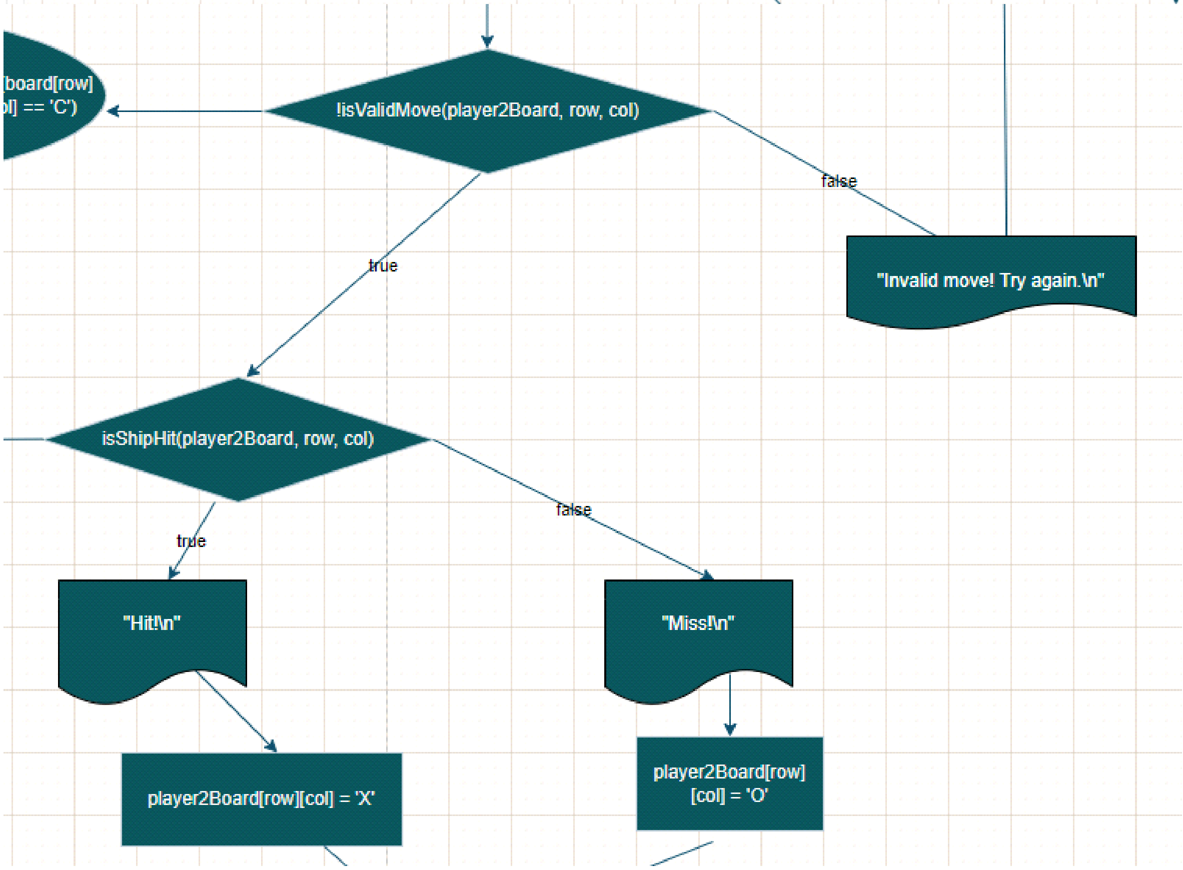

i++
false
i=0,
j
i < BOARD_SIZE
j=0
j < BOARD_SIZE
true
board[i][j] = '~'
j++

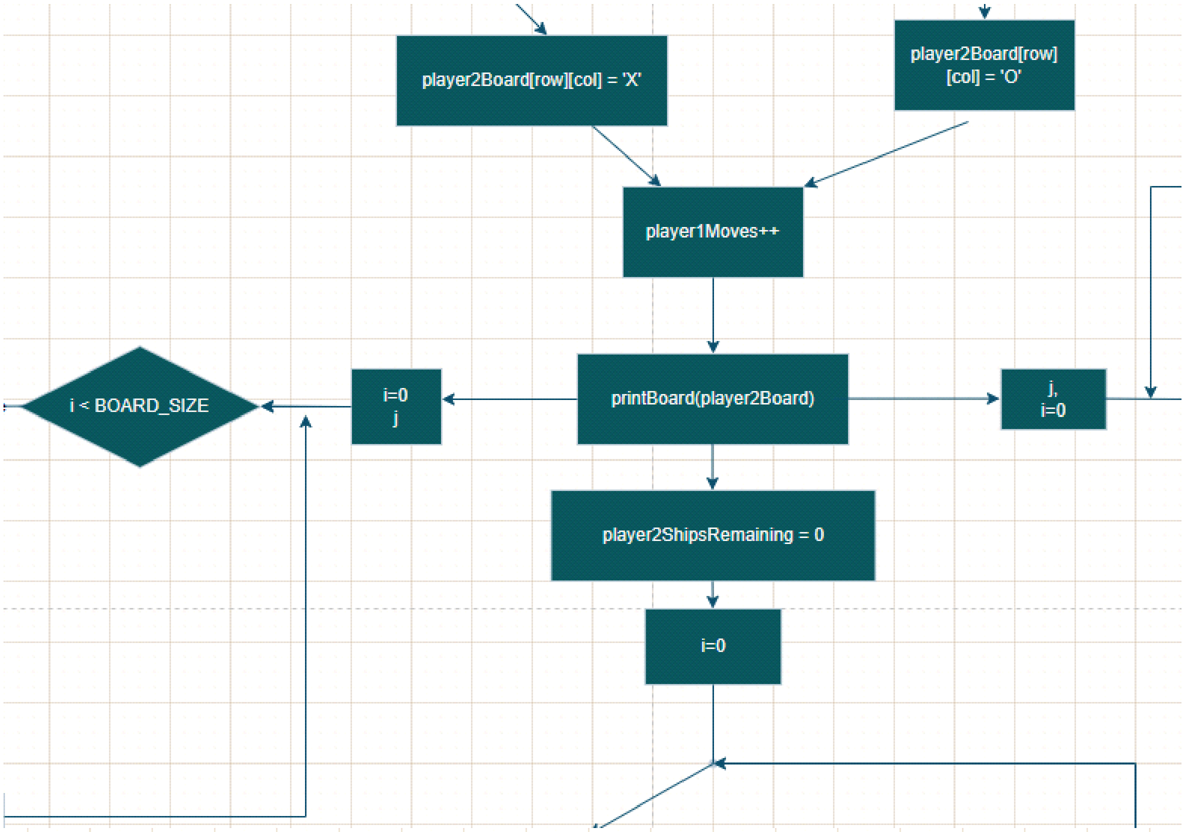

placeShips(player1Board)
placeShips(player2Board)
i=0
player1Moves = 0
player2Moves = 0
1
true
i, j

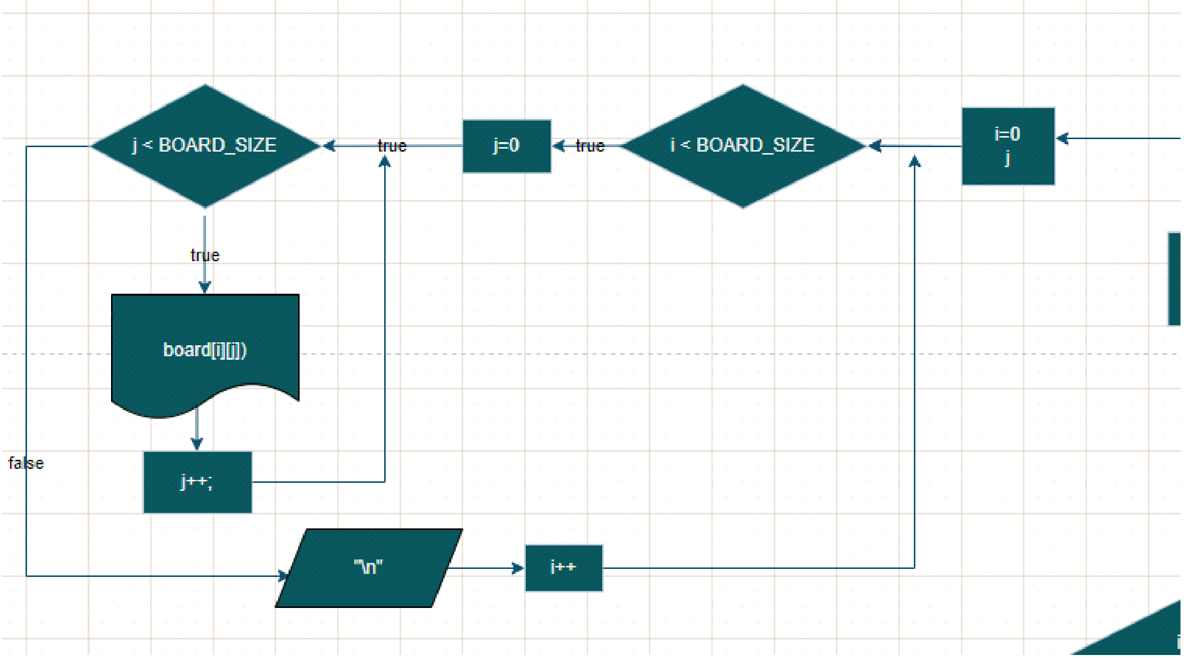

j++
i=0
SHIP_COUNT
true
row,col
false
row = rand() % BOARD_SIZE
col = rand() % BOARD_SIZE
!isValidMove(board, row, col)
true
shipSymbol = 'A' + i
board[row][col] = shipSymbol
i++

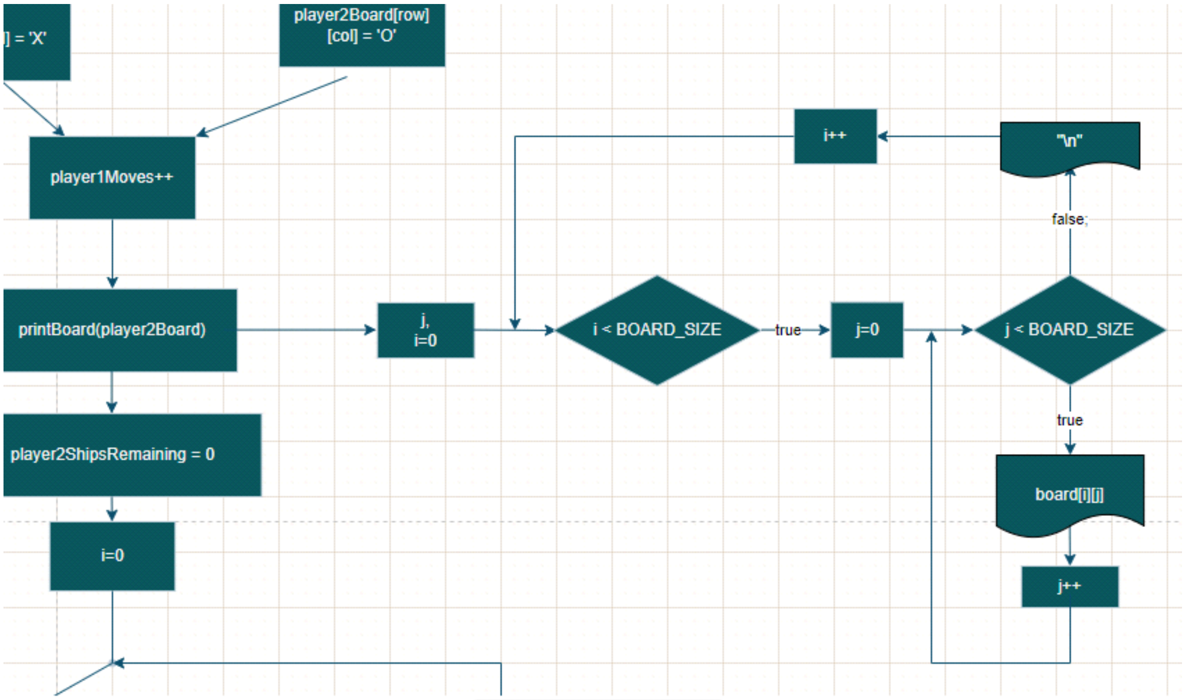

1
true
i, j
"Player 1's turn:\n"
"Enter row and column
(e.g., 1 2): "
row, col
row,col

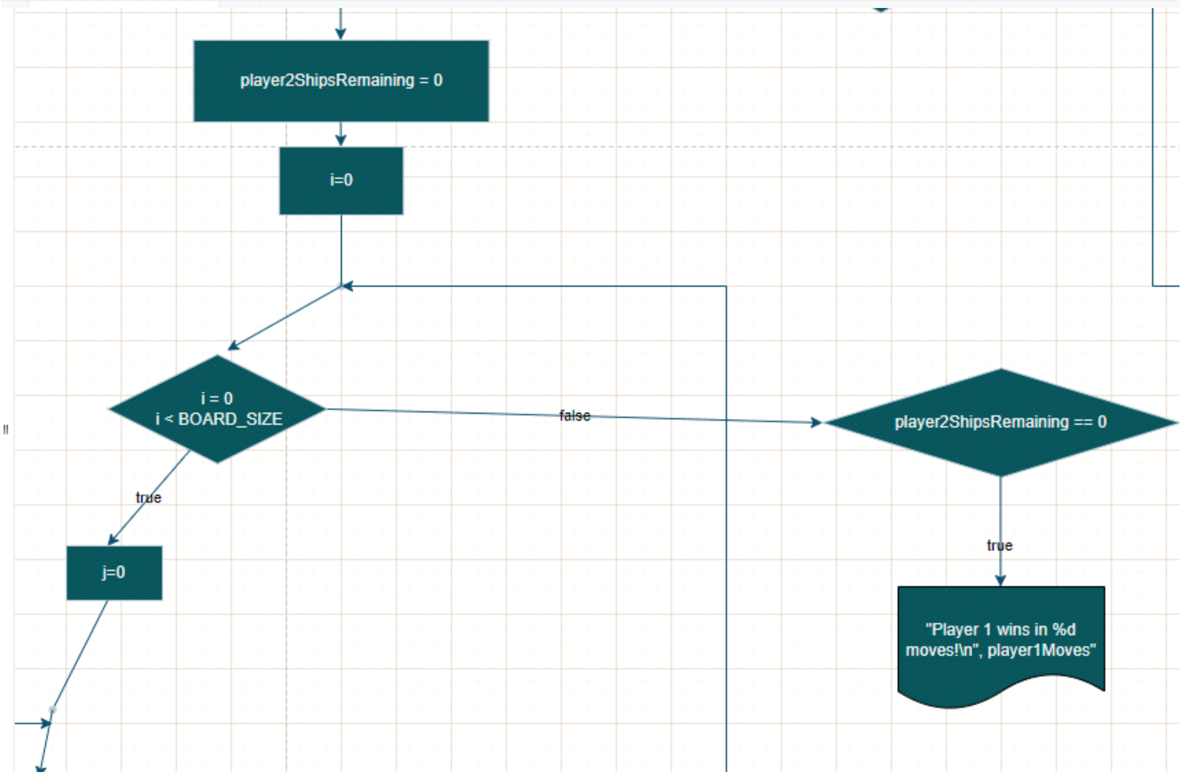

row,col
row >= 0 && row < BOARD_SIZE && col >= 0 && col < BOARD_SIZE && (board[row][col] == '~' || board[row][col] == 'A' || board[row][col] == 'B' || board[row][col] == 'C')
!isValidMove(player2Board, row, col)
true
return (board[row][col] != '~'
isShipHit(player2Board, row, col)
true
false
"Hit!\n"
"Miss!\n"

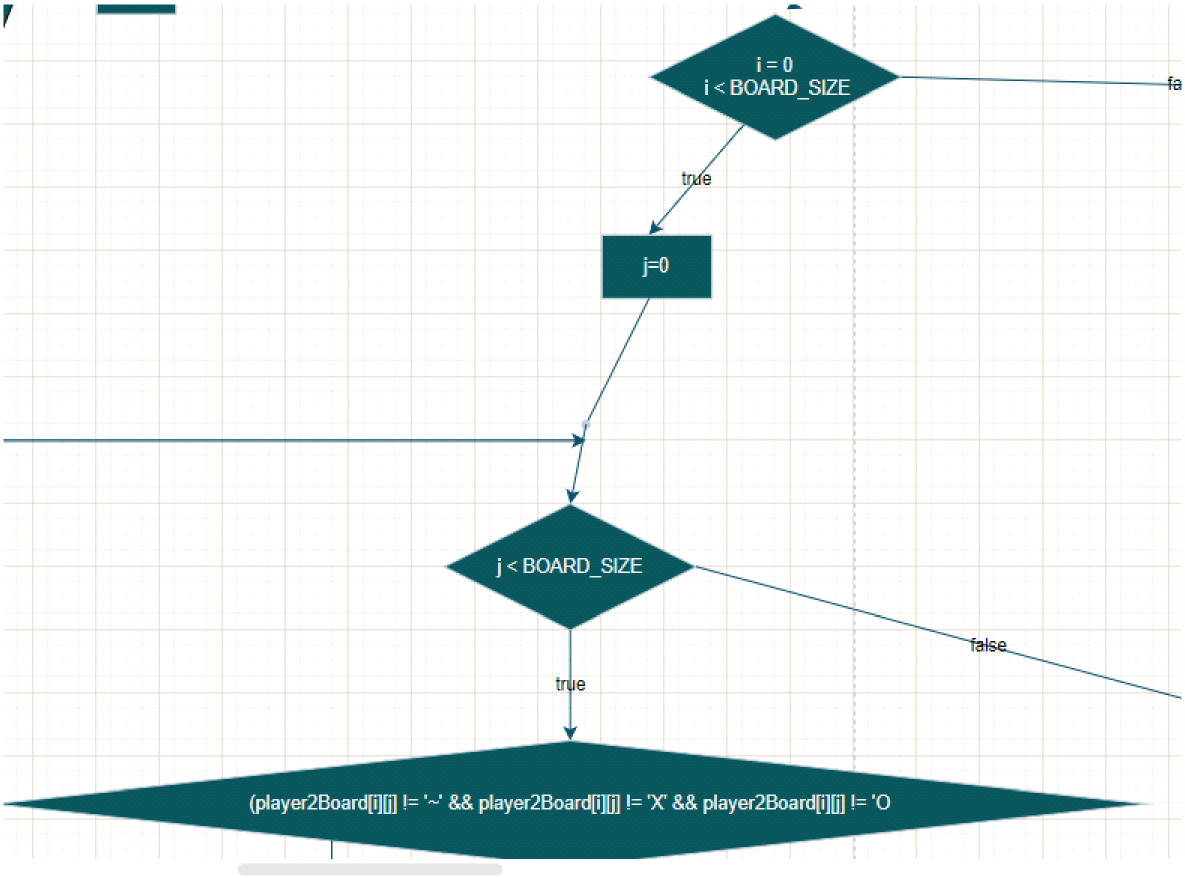

[board[row]
ol] == 'C')
!isValidMove(player2Board, row, col)
false
true
"Invalid move! Try again.\n"
isShipHit(player2Board, row, col)
true
false
"Hit!\n"
"Miss!\n"
player2Board[row][col] = 'X'
player2Board[row]
[col] = 'O'

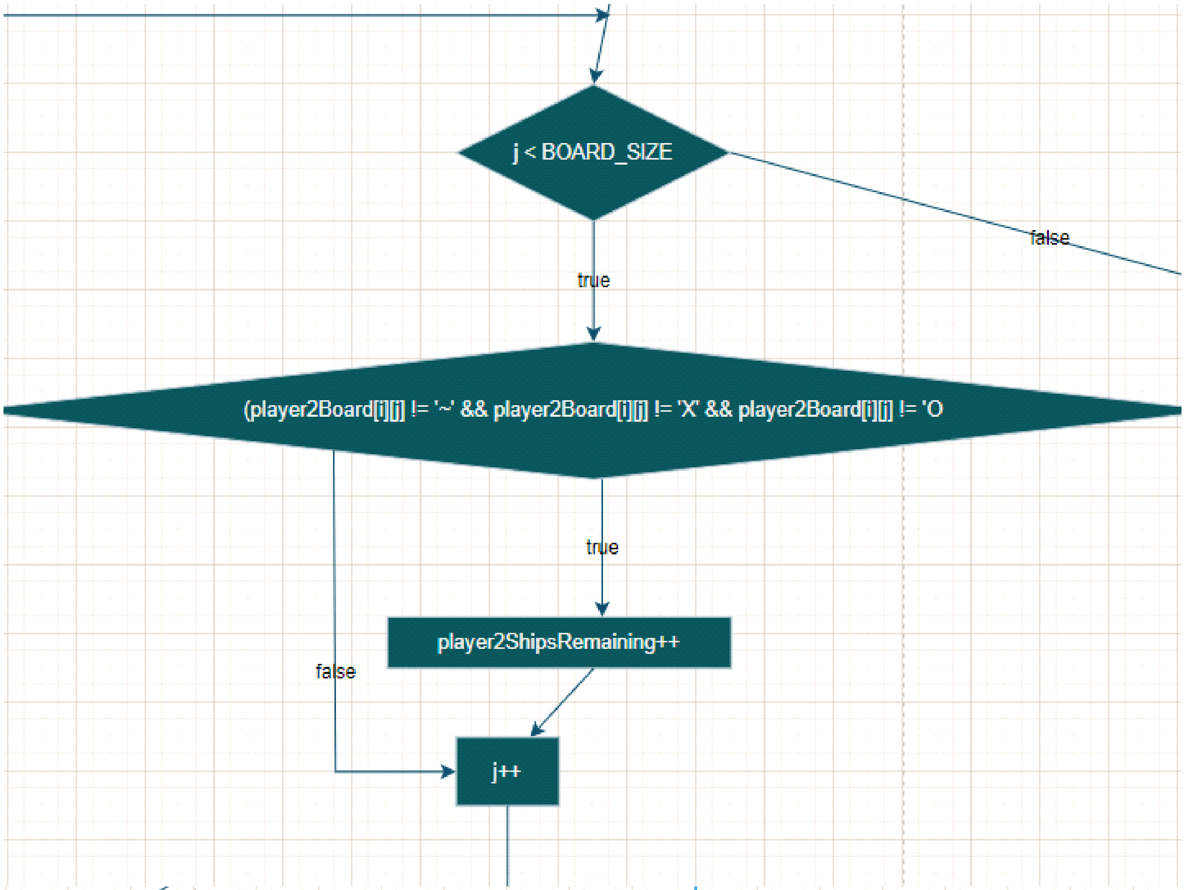

player2Board[row][col] = 'X'
player2Board[row] [col] = 'O'
player1Moves++
i < BOARD_SIZE
i=0 j
printBoard(player2Board)
j, i=0
player2ShipsRemaining = 0
i=0

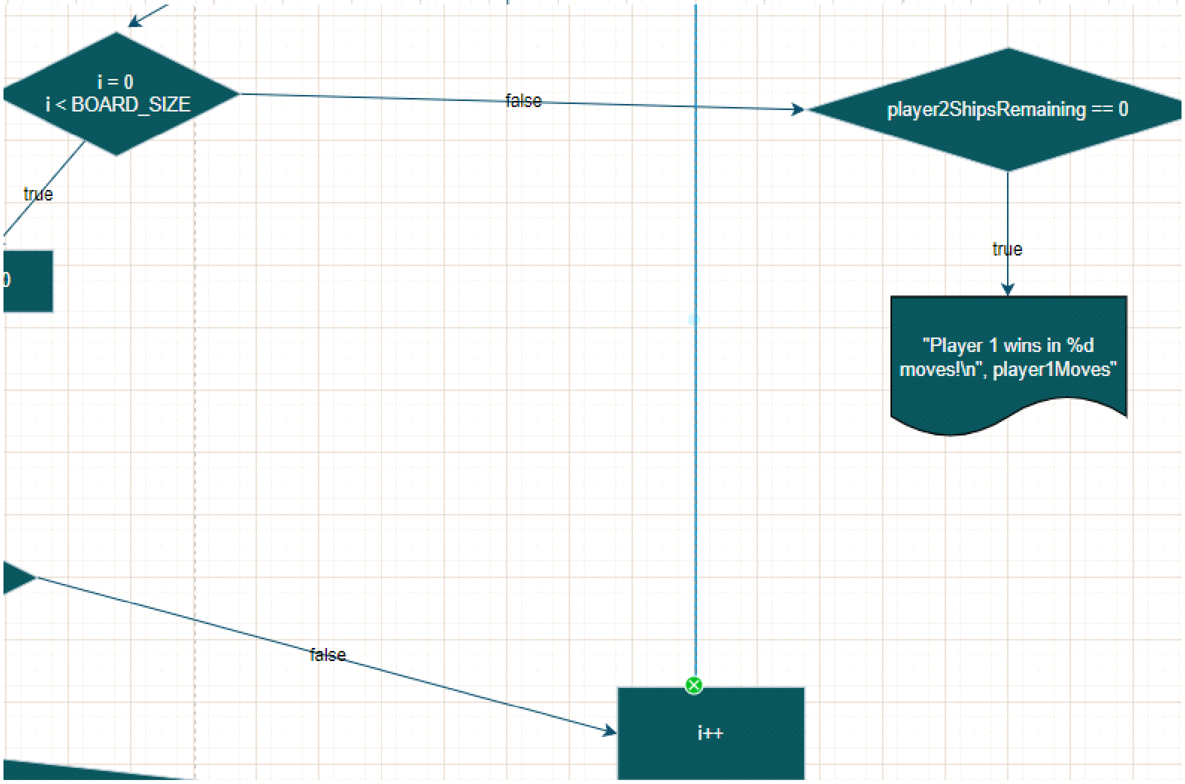

j < BOARD_SIZE
true
j=0
true
i < BOARD_SIZE
i=0 j
true
board[i][j])
false
j++;
"\n"
i++

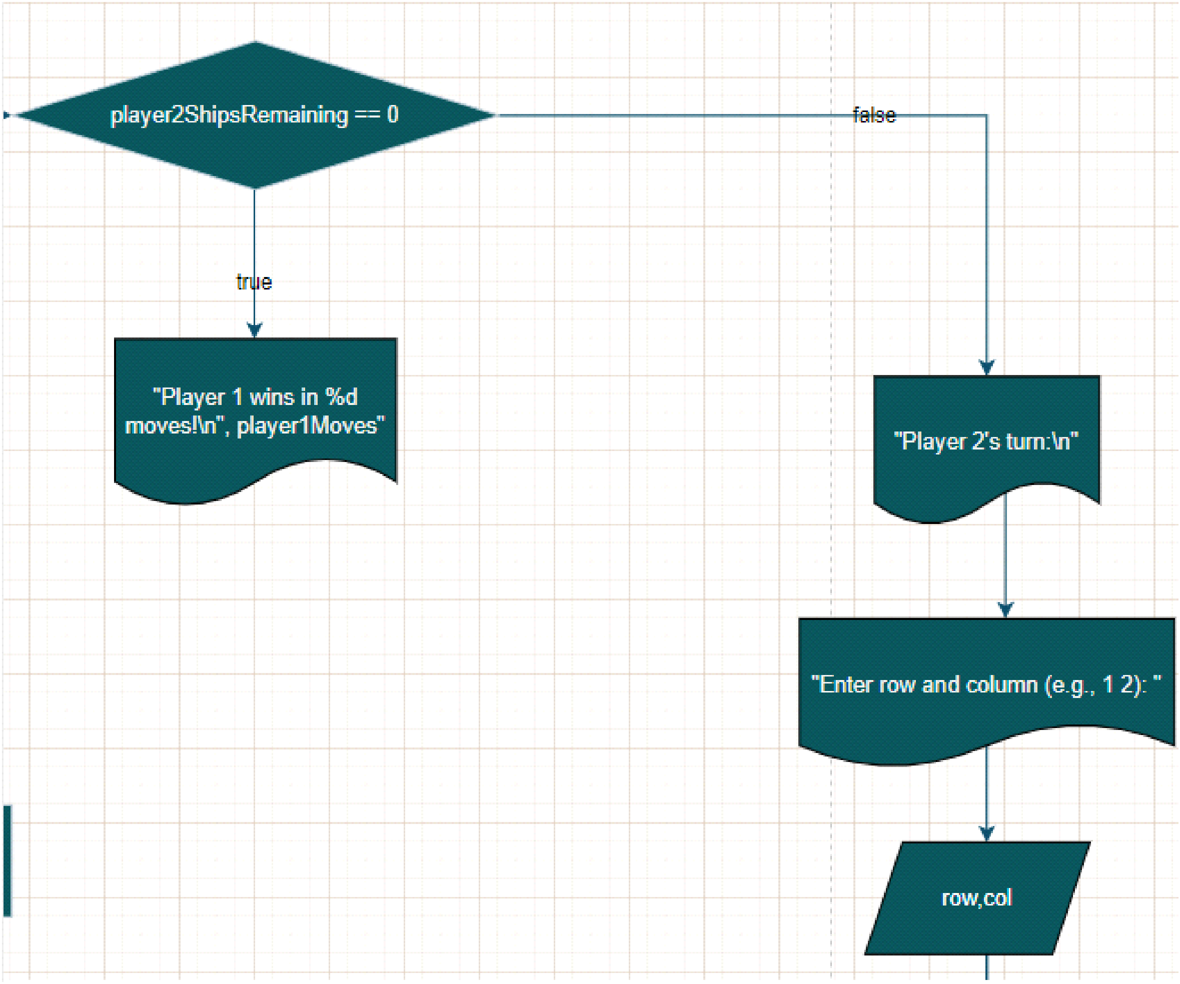

] = 'X'
player2Board[row]
[col] = 'O'
player1Moves++
i++
"\n"
false;
printBoard(player2Board)
j,
i=0
i < BOARD_SIZE
true
j=0
j < BOARD_SIZE
true
player2ShipsRemaining = 0
board[i][j]
i=0
j++

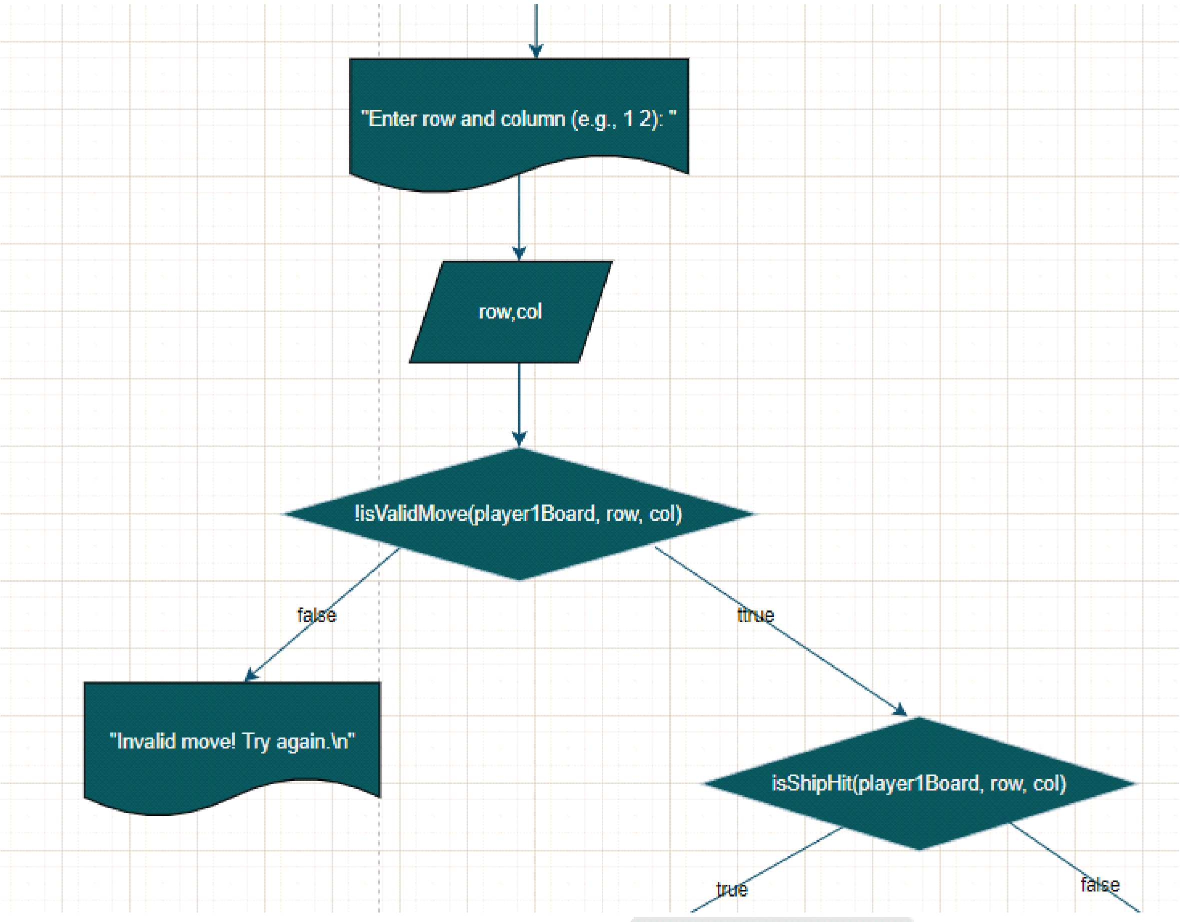

player2ShipsRemaining = 0
i=0
i = 0
i < BOARD_SIZE
false
player2ShipsRemaining == 0
true
j=0
true
"Player 1 wins in %d
moves!\n", player1Moves"

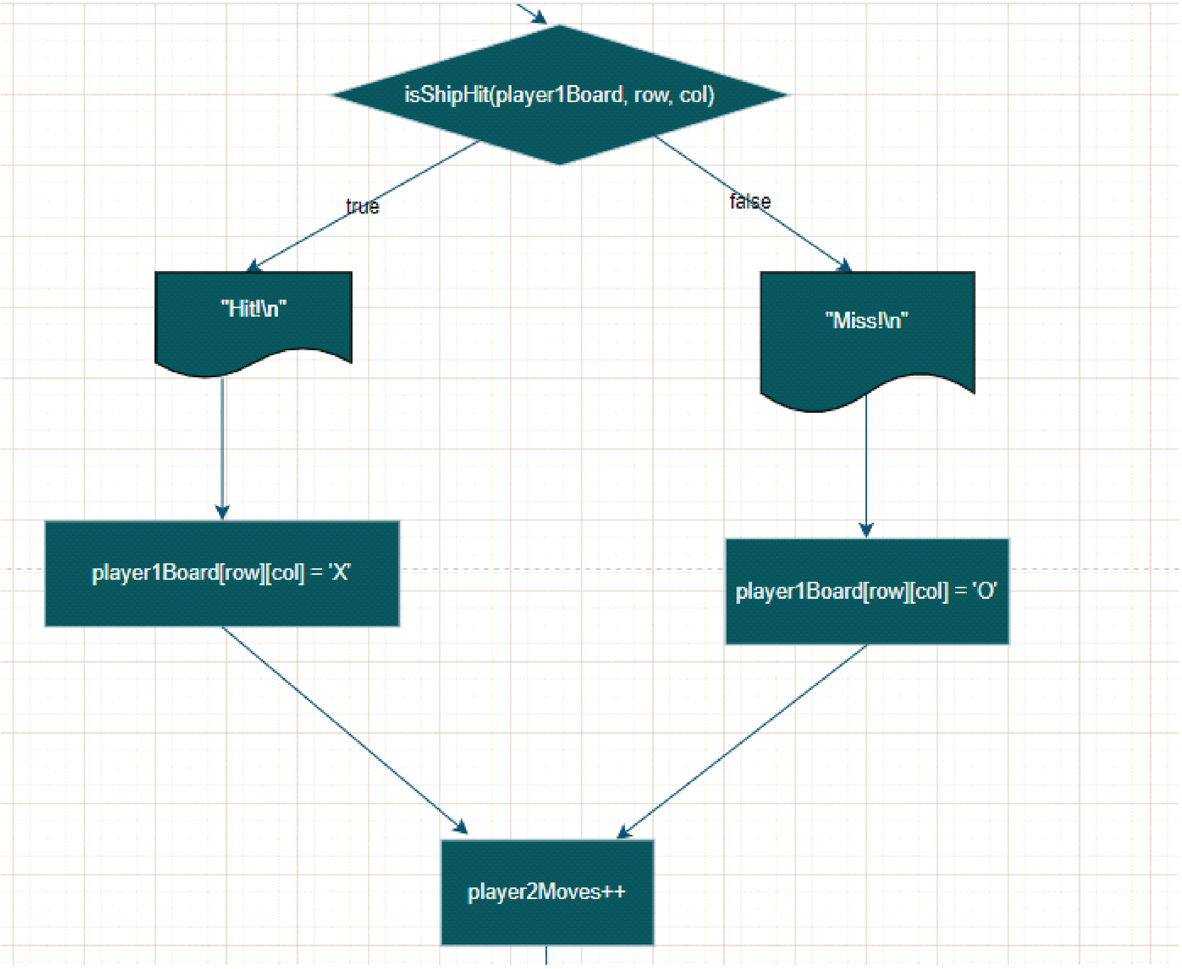

i = 0
i < BOARD_SIZE
fa
true
j=0
j < BOARD_SIZE
false
true
(player2Board[i][j] != '~' && player2Board[i][j] != 'X' && player2Board[i][j] != 'O

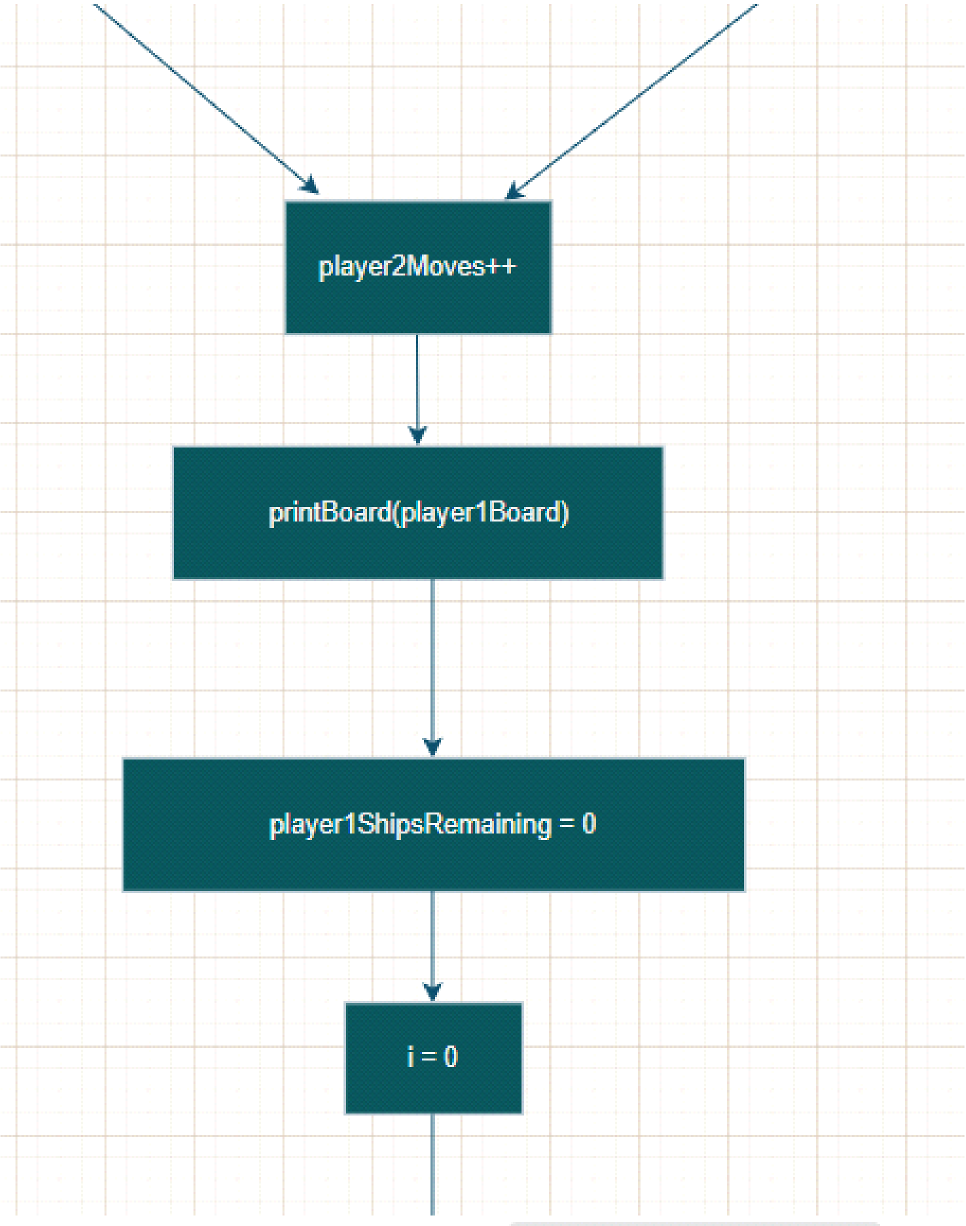

j < BOARD_SIZE
false
true
(player2Board[i][j] != '~' && player2Board[i][j] != 'X' && player2Board[i][j] != 'O
true
false
player2ShipsRemaining++
j++

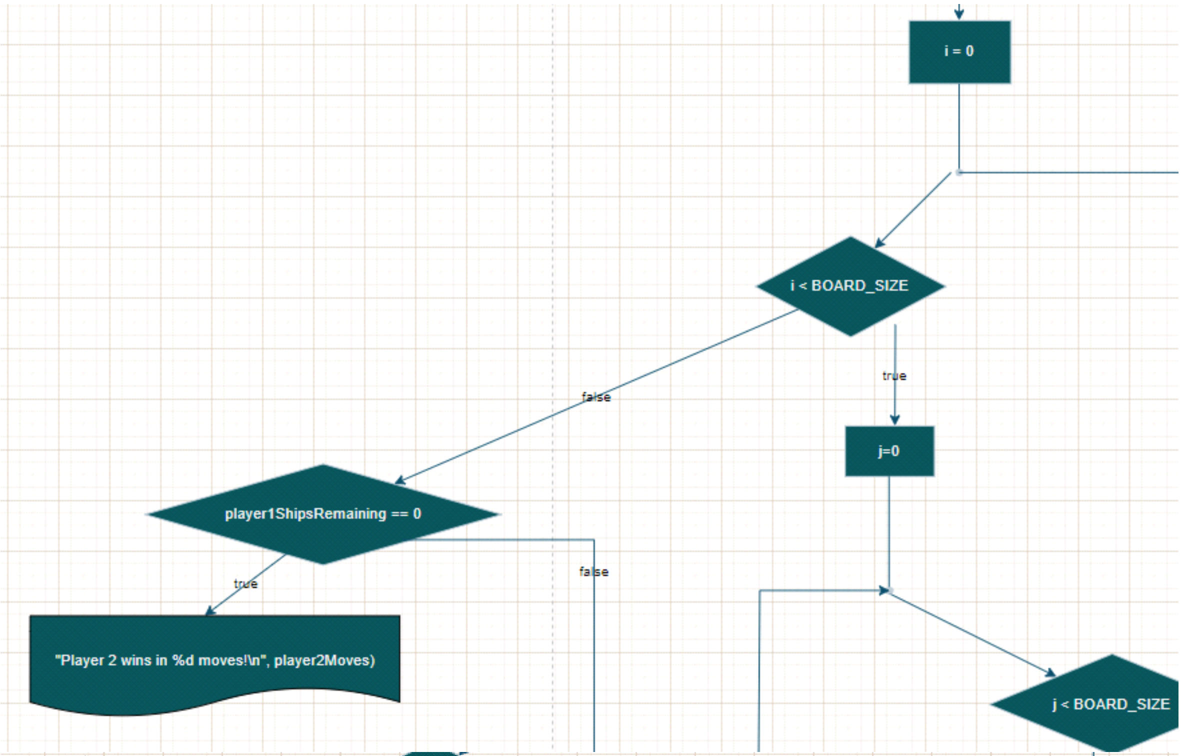

i = 0
i < BOARD_SIZE
false
player2ShipsRemaining == 0
true
true
0
"Player 1 wins in %d moves!\n", player1Moves"
false
i++

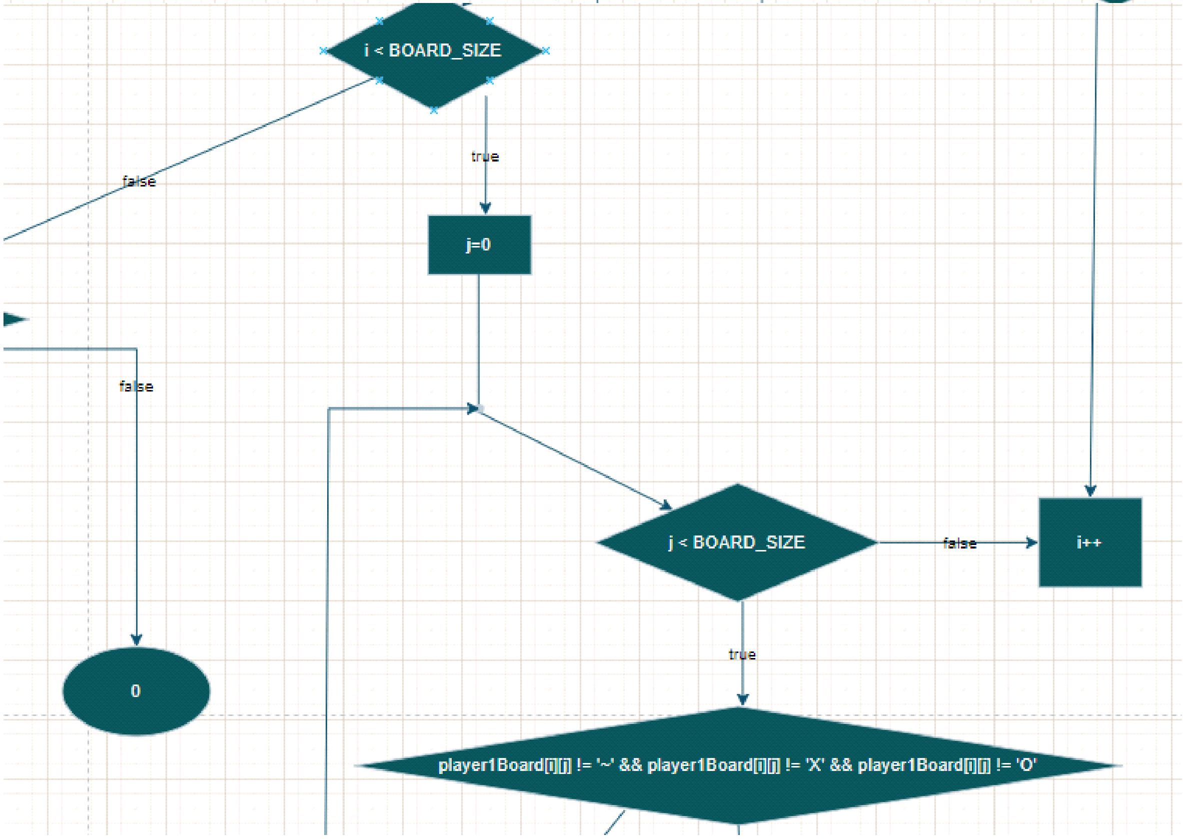

player2ShipsRemaining == 0
false
true
"Player 1 wins in %d moves!\n", player1Moves"
"Player 2's turn:\n"
"Enter row and column (e.g., 1 2): "
row,col

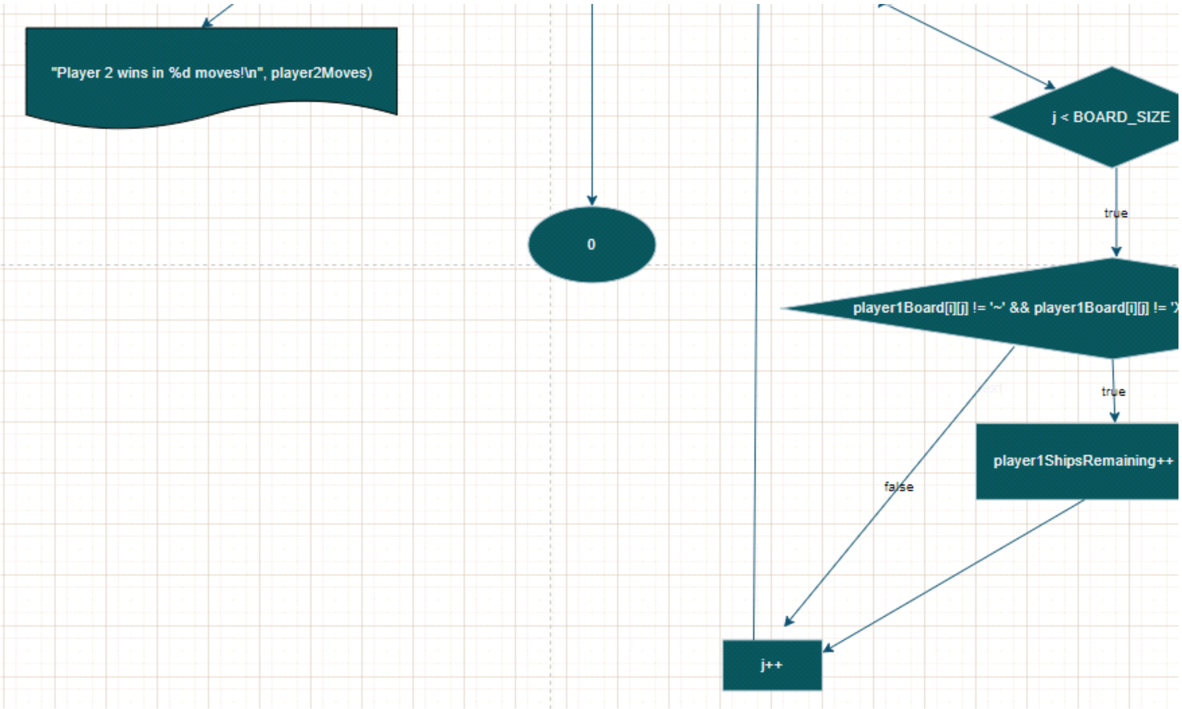

"Enter row and column (e.g., 1 2): "
row,col
!isValidMove(player1Board, row, col)
false
ttrue
"Invalid move! Try again.\n"
isShipHit(player1Board, row, col)
true
false


isShipHit(player1Board, row, col)
true
false
"Hit!\n"
"Miss!\n"
player1Board[row][col] = 'X'
player1Board[row][col] = 'O'
player2Moves++

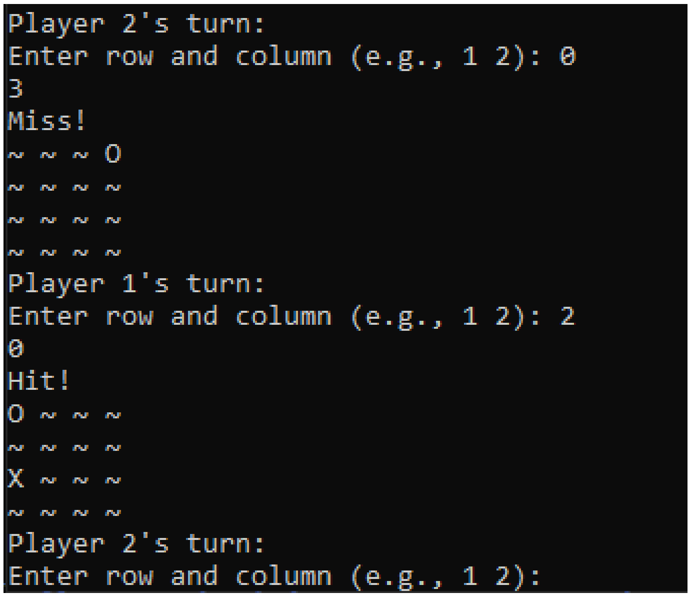

player2Moves++
printBoard(player1Board)
player1ShipsRemaining = 0
i = 0


i = 0
i < BOARD_SIZE
true
false
j=0
player1ShipsRemaining == 0
true
false
"Player 2 wins in %d moves!\n", player2Moves)
j < BOARD_SIZE

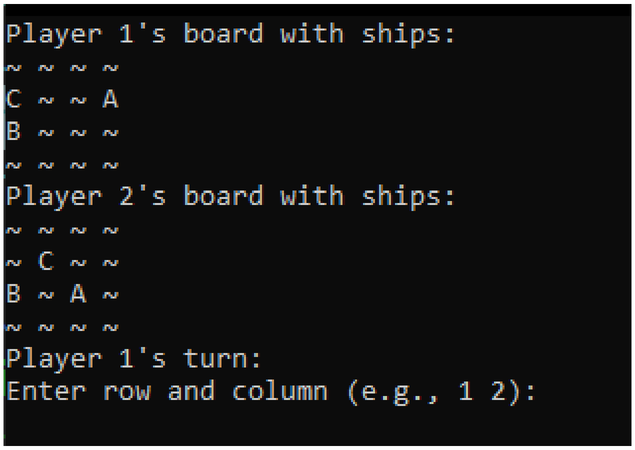

i < BOARD_SIZE
true
false
j=0
false
j < BOARD_SIZE
false
i++
true
0
player1Board[i][j] != '~' && player1Board[i][j] != 'X' && player1Board[i][j] != 'O'


"Player 2 wins in %d moves!\n", player2Moves)
j < BOARD_SIZE
true
0
player1Board[i][j] != '~' && player1Board[i][j] != 'X
true
player1ShipsRemaining++
false
j++

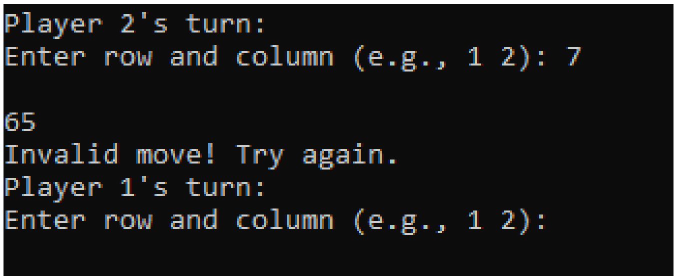
A few Screenshots of the Game



```
Player 2's turn:
Enter row and column (e.g., 1 2): 0
3
Miss!
~ ~ ~ O
~ ~ ~ ~
~ ~ ~ ~
~ ~ ~ ~
Player 1's turn:
Enter row and column (e.g., 1 2): 2
0
Hit!
O ~ ~ ~
~ ~ ~ ~
X ~ ~ ~
~ ~ ~ ~
Player 2's turn:
Enter row and column (e.g., 1 2):
```

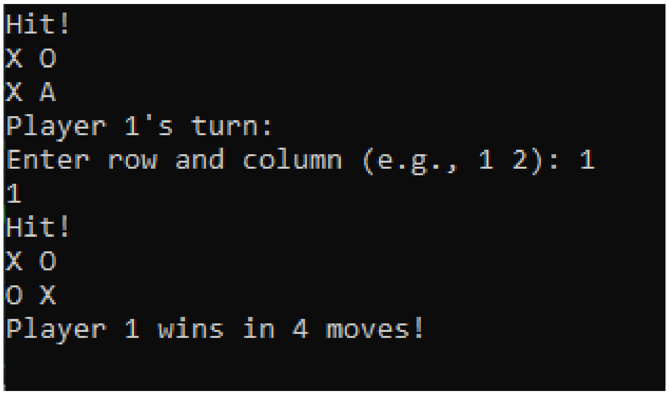


```
Player 1's board with ships:
~ ~ ~ ~
C ~ ~ A
B ~ ~ ~
~ ~ ~ ~
Player 2's board with ships:
~ ~ ~ ~
~ C ~ ~
B ~ A ~
~ ~ ~ ~
Player 1's turn:
Enter row and column (e.g., 1 2):
```



```
Player 2's turn:
Enter row and column (e.g., 1 2): 7

65
Invalid move! Try again.
Player 1's turn:
Enter row and column (e.g., 1 2):
```



```
Hit!
X O
X A
Player 1's turn:
Enter row and column (e.g., 1 2): 1
1
Hit!
X O
O X
Player 1 wins in 4 moves!
```

Source Code of the Game

```
#include <stdio.h>
#include <stdlib.h>
#include<stdbool.h>
#define BOARD_SIZE 10
#define SHIP_COUNT 3
//Oyun tablosunu olusturan fonksiyon
void initializeBoard(char board[BOARD_SIZE][BOARD_SIZE]) {
    int i,j;
    for (i = 0; i < BOARD_SIZE; i++) {
        for (j = 0; j < BOARD_SIZE; j++) {
            board[i][j] = '~';
        }
    }
}
//Oyun tablosunu ekrana yazdiran fonksiyon
void printBoard(char board[BOARD_SIZE][BOARD_SIZE]) {
    int i,j;
    for (i = 0; i < BOARD_SIZE; i++) {
        for (j = 0; j < BOARD_SIZE; j++) {
            printf("%c ", board[i][j]);
        }
        printf("\n");
    }
}
//Hamlenin gecerli olup olmadigini kontrol eden fonksiyon
int isValidMove(char board[BOARD_SIZE][BOARD_SIZE], int row, int col) {

    return (row >= 0 && row < BOARD_SIZE && col >= 0 && col < BOARD_SIZE &&
(board[row][col] == '~' || board[row][col] == 'A' || board[row][col] == 'B' || board[row][col]
== 'C'));

}
//Geminin vurulup vurulmadigini kontrol eden fonksiyon
int isShipHit(char board[BOARD_SIZE][BOARD_SIZE], int row, int col) {
    return (board[row][col] != '~');
}
//Gemileri oyun tablosuna rastgele yerlestiren fonksiyon

void placeShips(char board[BOARD_SIZE][BOARD_SIZE]) {
    int i;
    for (i = 0; i < SHIP_COUNT; i++) {
        int row, col;
        do {
            row = rand() % BOARD_SIZE;
            col = rand() % BOARD_SIZE;
        } while (!isValidMove(board, row, col));
        char shipSymbol = 'A' + i;

        board[row][col] = shipSymbol;
    }
}

int main() {
    char player1Board[BOARD_SIZE][BOARD_SIZE];
    char player2Board[BOARD_SIZE][BOARD_SIZE];

    initializeBoard(player1Board);
    initializeBoard(player2Board);

    placeShips(player1Board);
    placeShips(player2Board);

    printf("Player 1's board with ships:\n");
    printBoard(player1Board);

    printf("Player 2's board with ships:\n");
    printBoard(player2Board);

    int player1Moves = 0;
    int player2Moves = 0;

    while (1) {
        int i,j;
        // Oyuncu 1'in hamlesi
        printf("Player 1's turn:\n");
        printf("Enter row and column (e.g., 1 2): ");
        int row, col;
        scanf("%d %d", &row, &col);

        if (!isValidMove(player2Board, row, col)) {
            printf("Invalid move! Try again.\n");
            continue;
        }

        if (isShipHit(player2Board, row, col)) {
            printf("Hit!\n");
            player2Board[row][col] = 'X';
        } else {
            printf("Miss!\n");
            player2Board[row][col] = 'O';
        }

        player1Moves++;
        printBoard(player2Board);

        int player2ShipsRemaining = 0;
        //Oyuncunun kac tane gemisinin kaldıgini bulur
        //3 Gemisi de vurulan oyuncu oyunu kaybeder
        for (i = 0; i < BOARD_SIZE; i++) {

            for (j = 0; j < BOARD_SIZE; j++) {
                if (player2Board[i][j] != '~' && player2Board[i][j] != 'X' &&
player2Board[i][j] != 'O') {
                    player2ShipsRemaining++;
                }
            }
        }

        if (player2ShipsRemaining == 0) {
            printf("Player 1 wins in %d moves!\n", player1Moves);
            break;
        }

        // Oyuncu 2'nin hamlesi
        printf("Player 2's turn:\n");
        printf("Enter row and column (e.g., 1 2): ");
        scanf("%d %d", &row, &col);

        if (!isValidMove(player1Board, row, col)) {
            printf("Invalid move! Try again.\n");
            continue;
        }

        if (isShipHit(player1Board, row, col)) {
            printf("Hit!\n");
            player1Board[row][col] = 'X';
        } else {
            printf("Miss!\n");
            player1Board[row][col] = 'O';
        }

        player2Moves++;
        printBoard(player1Board);

        int player1ShipsRemaining = 0;

        for (i = 0; i < BOARD_SIZE; i++) {
            for (j = 0; j < BOARD_SIZE; j++) {
                if (player1Board[i][j] != '~' && player1Board[i][j] != 'X' && player1Board[i][j] != 'O') {
                    player1ShipsRemaining++;
                }
            }
        }

        if (player1ShipsRemaining == 0) {
            printf("Player 2 wins in %d moves!\n", player2Moves);
            break;
        }
    }

    return 0;
}
```