Determine the winner in Tic-Tac-Toe

The first thing to think about is the address of each cell. Since they are organized as an array, cells are addressed by the name of the array and the row and column. When the game is being played, the user enters 1-9 to select a square. An 'X' or 'O' is placed in the square. Internal to the program, the game board is stored in a 3 by 3 two dimensional array,

```
char board[3][3] = {    // index for positions in the array
    {'1', '2', '3'},        //  [0][0]  [0][1]  [0][2]
    {'4', '5', '6'},        //  [1][0]  [1][1]  [1][2]
    {'7', '8', '9'}         //  [2][0]  [2][1]  [2][2]
};
```

Data can be placed in a cell or retrieved using the array name and its indexes. Example: board[row][column]
where **board** is the name of the array
row is the row identifier (0, 1 or 2)
column is the column identifier (0, 1 or 2)

Shown here are the initial contents of each cell and the cell address

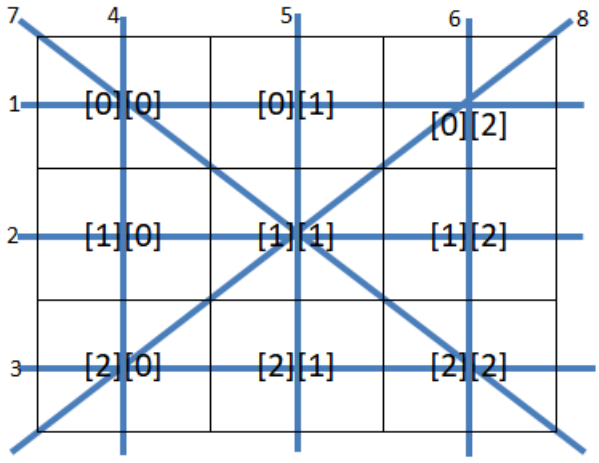
	Colum 0	Column 1	Column 2
Row 0	1 [0][0]	2 [0][1]	3 [0][2]
Row 1	4 [1][0]	5 [1][1]	6 [1][2]
Row 2	7 [2][0]	8 [2][1]	9 [2][2]

Row 0 is the top row
Row 1 is the middle row
Row 2 is the bottom row

Column 0 is the left column
Column 1 is the center column
Column 2 is the right column

Cells 1, 5 and 9 are the left diagonal \
Cells 3, 5 and 7 are the right diagonal /

There are eight ways that the game can be won with three in a row (or column, or at either angle \ /)



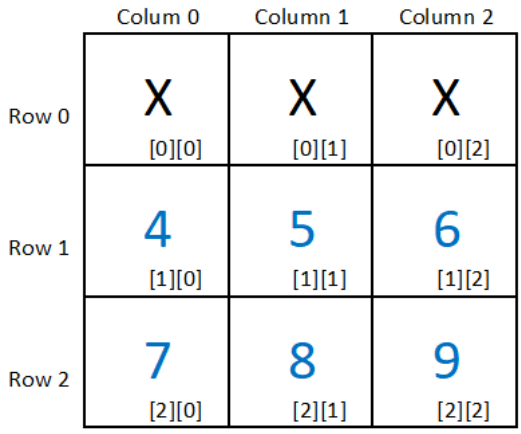
1. [0][0] [0][1] [0][2] Across the top —
2. [0][1] [1][1] [1][2] Across the middle —
3. [2][0] [2][1] [2][2] Across the bottom —
4. [0][0] [1][0] [2][0] Down the left |
5. [0][1] [1][1] [2][1] Down the center |
6. [0][2] [1][2] [2][2] Down the right |
7. [0][0] [1][1] [2][2] Diagonal from the left \
8. [0][2] [1][1] [2][0] Diagonal from the right /

If any set of three squares has the same value, either 'X' or 'O' then there is a winner.
The **CheckForWinningGame** function should return the winner, either 'X' or 'O' or '-' if no winner.
An easy way to check for a winner is to have eight **if** statements. Set the **player** variable to the value in the first set of three boxes and then see if it is the same as the other two boxes in the set.

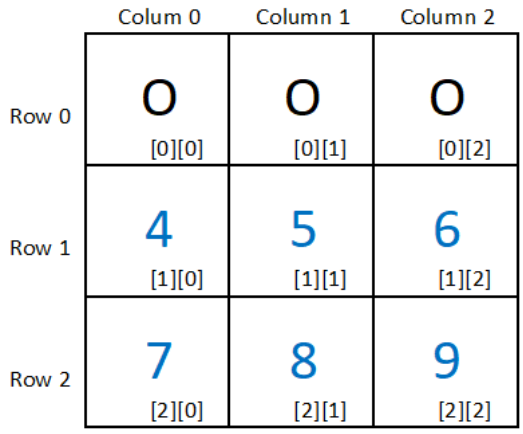
Test the top row for a winner by checking to see if the second cell is the same as the first cell AND the third cell is the same as the first cell. If they are all the same, return the contents of the first cell as the winner.

```
char player;

// The addresses for the top row are:  [0][0],  [0][1]  and  [0][2]
// set player =  the first cell on the top row, then see if it matches the others on the top row
player = board[0][0];
if (  board[0][1] == player   &&   board[0][2] == player )   return player;  // the winner
```



Three X's in a row, return board[0][0] // which is 'X'



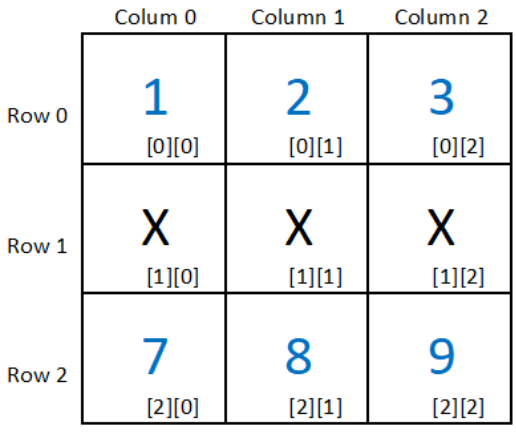
Three O's in a row, return board[0][0] // which is 'O'

==========

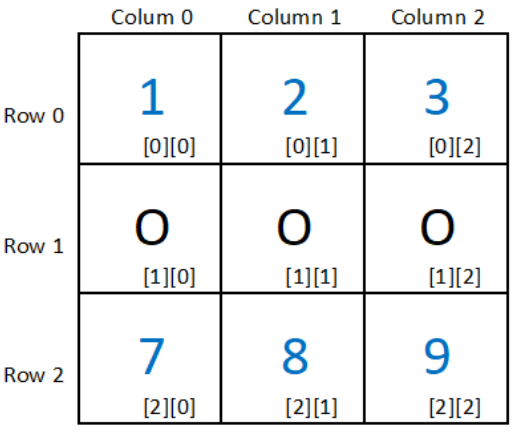
Now it is time to test the second row for a winner.

// char player; is already declared, so it does not need to be declared again

```
// The addresses for the middle row are:  [1][0],  [1][1]  and  [1][2]
// set player =  the first cell on the middle row, then see if it matches the others the middle row
player = board[1][0];
if (  board[1][1] == player   &&   board[1][2] == player )   return player;  // the winner
```



Three X's in a row, return board[0][0] // which is 'X'



Three O's in a row, return board[0][0] // which is 'O'

==========

Checking the third row is similar to the first row, but using addresses [2][0], [2][1] and [2][2]
Checking the columns is similar to rows, but use cell addresses going down instead of across
Addresses for the first column are [0][0], [1][0] and [2][0]. You should be able to figure out the rest.
Checking the diagonals is also similar. Use the cell addresses in a diagonal \ and then cell addresses for /
If you reach the end of all of the **if** statements, then a winner has not been found. Return a '-' to indicate no winner.

```
// ------ Check all eight possible combinations for a win.
// return the player ('X' or 'O') if the game has been won
// return '-' if there is currently no winner
char CheckForWinningGame(char board[3][3])
{
    char player;
    // check going across the top row to see if all squares are the same
    player = board[0][0];  // start with the upper left corner square
    if (board[0][1] == player && board[0][2] == player) return player;

    // check going across the middle row to see if all squares are the same
    player = board[0][1];  // start with the left middle square
    if (board[1][1] == player && board[1][2] == player) return player;

    //
    // continue checking the other six combinations for a winning game by setting
    //     player to the first square in the set of three squares and then
    //     using an if statement with its return statement to see if the contents
    //     of the other two squares are the same as the first square in the set
    //

    // There is no winner if none of the eight combinations passed an  if  statement
    return '-';    // return '-' to indicate that the game has not yet been won

} // end of the CheckForWinningGame function
```

There should be eight **if** statements that have a return. These will cause the **CheckForWinningGame** to return either the 'X' or 'O' that was three in a row (or column or angle \ /). When none of the **if** statements execute their **return player;** statement, then a winner was not detected. The program should have the **return '-';** statement as the last thing that happens before the closing curly-brace } that ends the **CheckForWinningGame** function.