4-Digit Code Breaker Game

ReadMe

Name: Edzhe Veli

ID: 001501763

Course ID: COMP1950

Lecturer: Fazle Chowdhury

Table of Contents

1. Project Overview

This 4 digit code breaker game also known as a mastermind game is created with HTML, CSS, and JavaScript that allows player to play the game until they reach their 10th attempt. After each guess, the player will receive information about players guesses in the form of coloured pegs;

Black peg= a digit is correctly placed in the correct position.

White peg= a digit is correctly placed in an incorrect position.

The game also has a scoring system, and it can persistently store the players highest score in the local storage for future reference. The player can save a game at any time and resume it when he/she wants.

2. How To Run

1. Extract all files from the ZIP folder to a local folder.
2. Ensure all files index.html, styles.css and script.js are in the same folder.
3. Click twice on index.html.
4. The game will load immediately it doesn't require to any installation.
5. Supported browsers: Chrome, Safari. Edge.

3. How To Play

1. Start: Game automatically generates 4 digit code
2. Guess: Click the number buttons to build your 4 digit guess
3. Submit: Click “Check Guess” to submit your attempt
4. Feedback: black peg – correct digit correct position / white peg - correct digit wrong position.
5. Refine: You can use the feedback to make successful decisions.
6. To win: Crack the code within 10 attempts.

4. Features

Game Features;

- Random code generation
- Interactive digit selections with clickable buttons
- Visual feedbacks with black and white pegs
- Maximum 10 attempts
- Automatic detection for winning or losing
- Secret code reveal end of the game
- Top 10 highest scores are sorted and saved into the localStorage
- Save current game to play later

User Interface:

- Clean and responsive design
- How to play instructions section
- Visual attempts history
- Score display
- Reset and save buttons

5. File Structure

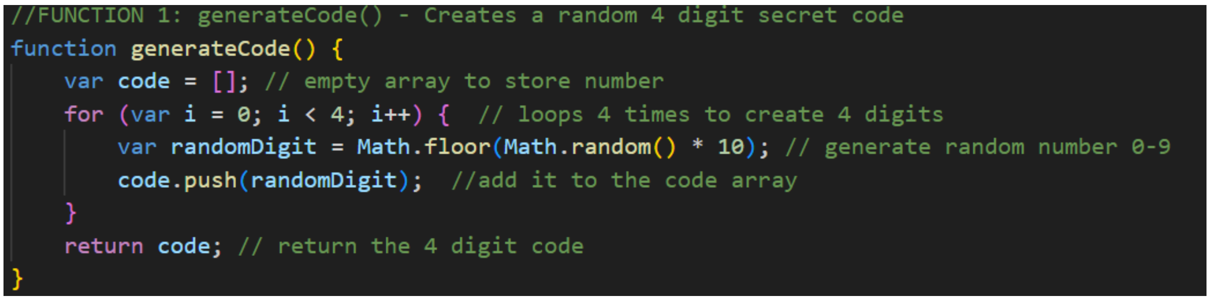
Index.html, ~150, lines contains HTML structure (game board, controls, modals)

Styles.css, ~400, lines styling and all styling organized with comments.

Script.js, ~550 lines, game logic and interactive functions.

6. Detailed Function Documentation

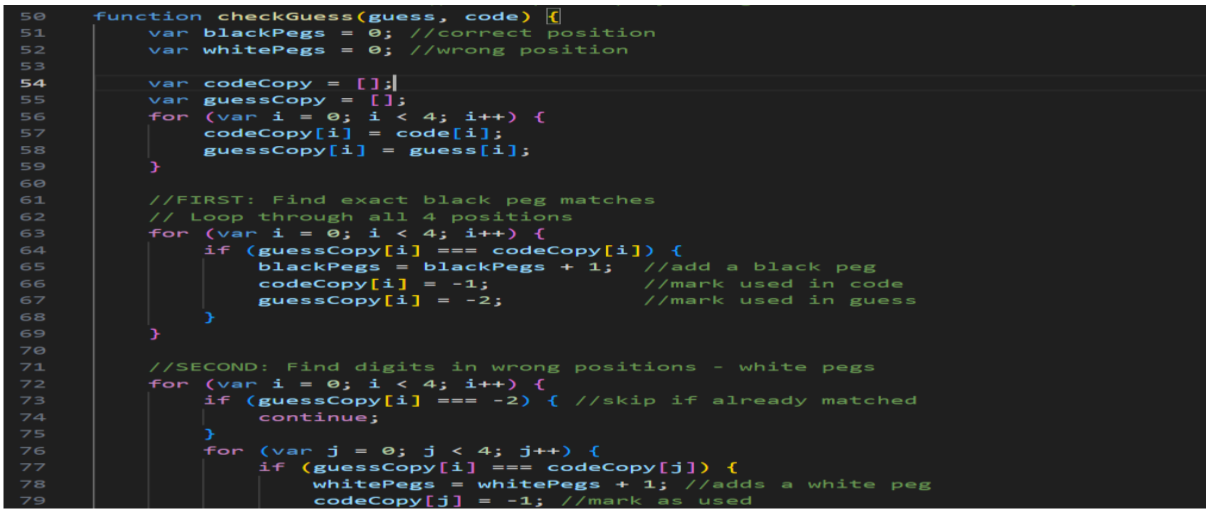
6.1 generateCode() - creates random 4 digit secret code at the start of each game.



```
//FUNCTION 1: generateCode() - Creates a random 4 digit secret code
function generateCode() {
    var code = []; // empty array to store number
    for (var i = 0; i < 4; i++) {  // loops 4 times to create 4 digits
        var randomDigit = Math.floor(Math.random() * 10); // generate random number 0-9
        code.push(randomDigit);  //add it to the code array
    }
    return code; // return the 4 digit code
}
```

- Creates an empty array to hold the generated code.
- Iterates through for loop 4 times.
- Generates a random number using Math.random().
- Converts that random number into an integer while using Math.floor(math.random) *10).
- Pushes each of these digits into the array.
- Returns the entire array with all 4 digits.

6.2 checkGuess(guess,code) - compares the players guess against the secret code



```
function checkGuess(guess, code) {
    var blackPegs = 0; //correct position
    var whitePegs = 0; //wrong position

    var codeCopy = [];
    var guessCopy = [];
    for (var i = 0; i < 4; i++) {
        codeCopy[i] = code[i];
        guessCopy[i] = guess[i];
    }

    //FIRST: Find exact black peg matches
    // Loop through all 4 positions
    for (var i = 0; i < 4; i++) {
        if (guessCopy[i] === codeCopy[i]) {
            blackPegs = blackPegs + 1;   //add a black peg
            codeCopy[i] = -1;            //mark used in code
            guessCopy[i] = -2;           //mark used in guess
        }
    }

    //SECOND: Find digits in wrong positions - white pegs
    for (var i = 0; i < 4; i++) {
        if (guessCopy[i] === -2) { //skip if already matched
            continue;
        }
        for (var j = 0; j < 4; j++) {
            if (guessCopy[i] === codeCopy[j]) {
                whitePegs = whitePegs + 1; //adds a white peg
                codeCopy[j] = -1; //mark as used
```

Algorithm Explanation:

Creates copies of both arrays to avoid modifying originals.

First section: Increment black pegs whenever a digit is present in the same index box.

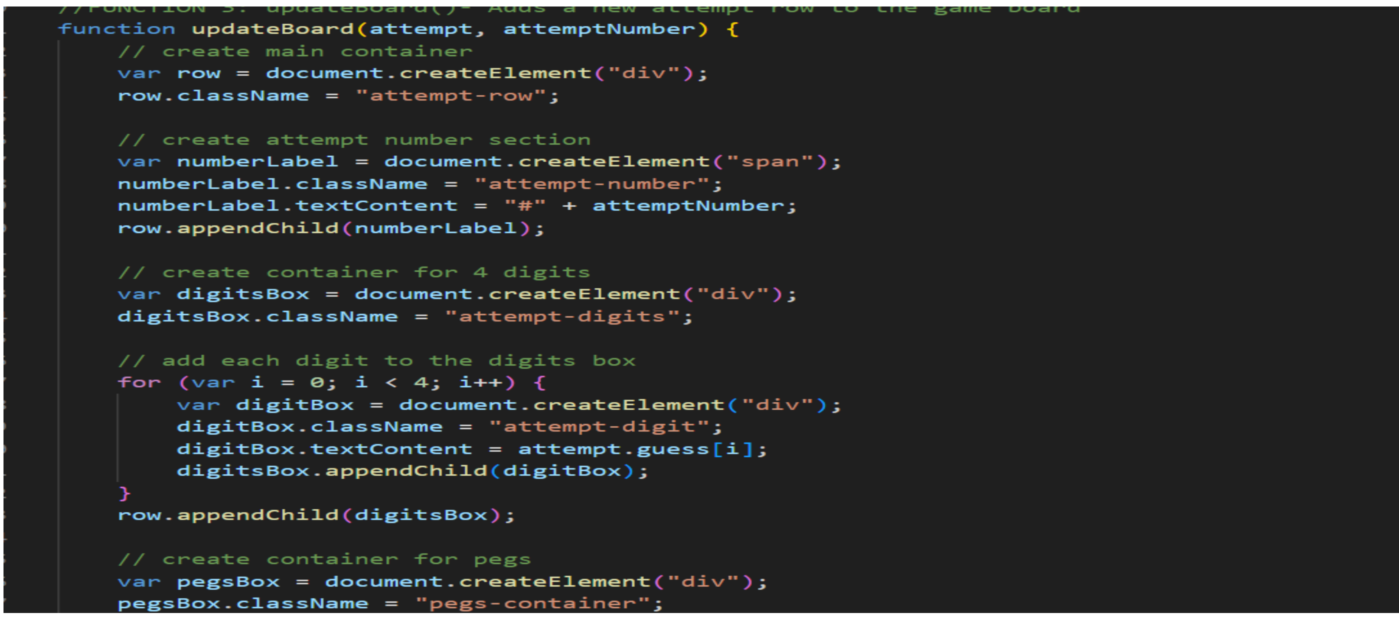
Flag each of these locations with a unique value to prevent double counting.

Second section: Do not checks digits which have already been matched.

Checks the remaining digits in the code copy.

Finds the digit in the code copy, if found, increment white pegs and flag it as used.

6.3 updateBoard – dynamically updates the DOM to display the players attempt



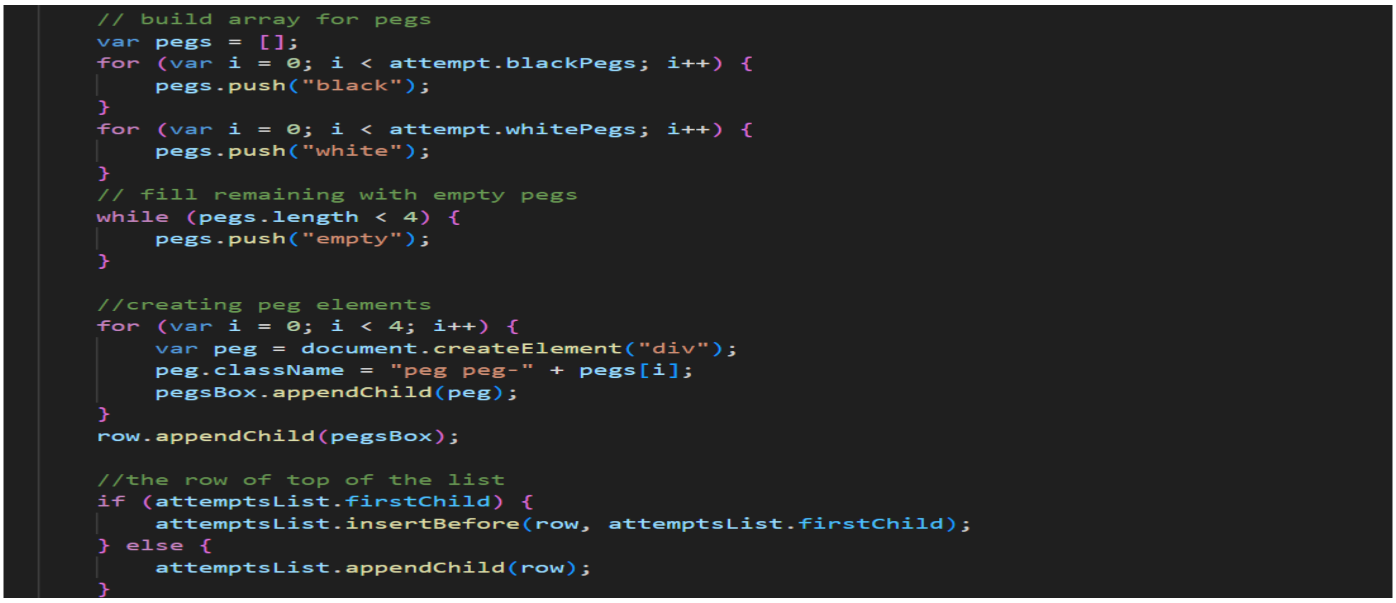
```
//FUNCTION 3: updateBoard()- Adds a new attempt row to the game board
function updateBoard(attempt, attemptNumber) {
    // create main container
    var row = document.createElement("div");
    row.className = "attempt-row";

    // create attempt number section
    var numberLabel = document.createElement("span");
    numberLabel.className = "attempt-number";
    numberLabel.textContent = "#" + attemptNumber;
    row.appendChild(numberLabel);

    // create container for 4 digits
    var digitsBox = document.createElement("div");
    digitsBox.className = "attempt-digits";

    // add each digit to the digits box
    for (var i = 0; i < 4; i++) {
        var digitBox = document.createElement("div");
        digitBox.className = "attempt-digit";
        digitBox.textContent = attempt.guess[i];
        digitsBox.appendChild(digitBox);
    }
    row.appendChild(digitsBox);

    // create container for pegs
    var pegsBox = document.createElement("div");
    pegsBox.className = "pegs-container";
```



```
    // build array for pegs
    var pegs = [];
    for (var i = 0; i < attempt.blackPegs; i++) {
        pegs.push("black");
    }
    for (var i = 0; i < attempt.whitePegs; i++) {
        pegs.push("white");
    }
    // fill remaining with empty pegs
    while (pegs.length < 4) {
        pegs.push("empty");
    }

    //creating peg elements
    for (var i = 0; i < 4; i++) {
        var peg = document.createElement("div");
        peg.className = "peg peg-" + pegs[i];
        pegsBox.appendChild(peg);
    }
    row.appendChild(pegsBox);

    //the row of top of the list
    if (attemptsList.firstChild) {
        attemptsList.insertBefore(row, attemptsList.firstChild);
    } else {
        attemptsList.appendChild(row);
    }
```

DOM Manipulation:

-Uses document.getElementById() to find the container element.

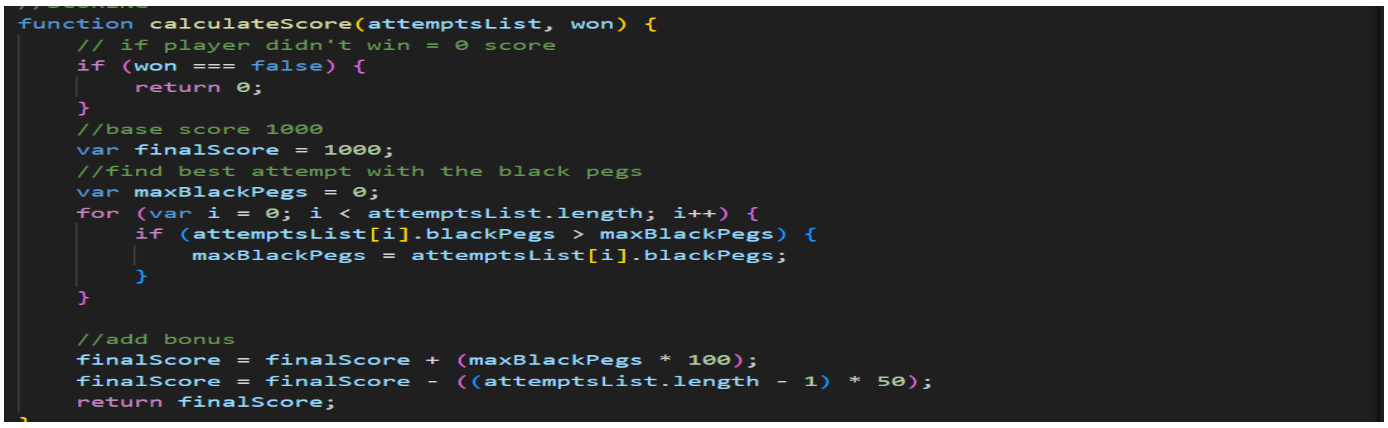
-It uses document.createElement() to create new HTML elements.

-It uses appendChild() to add elements to the DOM tree.

-CSS are assigned with using className .

-Text content is uses textContent.

6.4 calculateScore – Calculates the players final score



```javascript
function calculateScore(attemptsList, won) {
    // if player didn't win = 0 score
    if (won === false) {
        return 0;
    }
    //base score 1000
    var finalScore = 1000;
    //find best attempt with the black pegs
    var maxBlackPegs = 0;
    for (var i = 0; i < attemptsList.length; i++) {
        if (attemptsList[i].blackPegs > maxBlackPegs) {
            maxBlackPegs = attemptsList[i].blackPegs;
        }
    }

    //add bonus
    finalScore = finalScore + (maxBlackPegs * 100);
    finalScore = finalScore - ((attemptsList.length - 1) * 50);
    return finalScore;
```

Scoring Formula:

-Base score: 1000 points for win

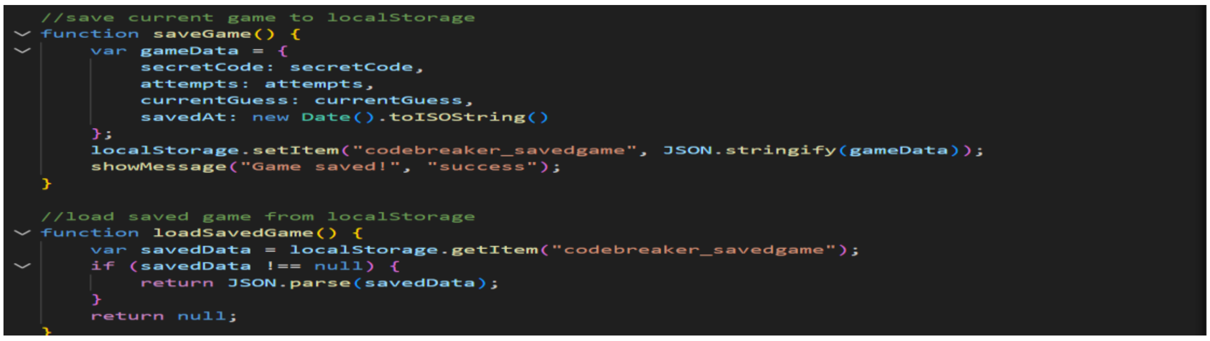
-Bonus: +100 points per maximum sequential correct digit

-Minimum score is 100

6.5 localStorage Functions

- saveGame()

-loadSaveGame()



```
//save current game to localStorage
function saveGame() {
    var gameData = {
        secretCode: secretCode,
        attempts: attempts,
        currentGuess: currentGuess,
        savedAt: new Date().toISOString()
    };
    localStorage.setItem("codebreaker_savedgame", JSON.stringify(gameData));
    showMessage("Game saved!", "success");
}

//load saved game from localStorage
function loadSavedGame() {
    var savedData = localStorage.getItem("codebreaker_savedgame");
    if (savedData !== null) {
        return JSON.parse(savedData);
    }
    return null;
}
```

LocalStorage Explanation:

-localStorage.setItem(key,value) - stores data in to the browser

-localStorage.getItem(key) - retrieves data from browser

-JSON.parse()- converts string back to JS object

-Data persist even after browser closed

7. Arrays and Loops Used

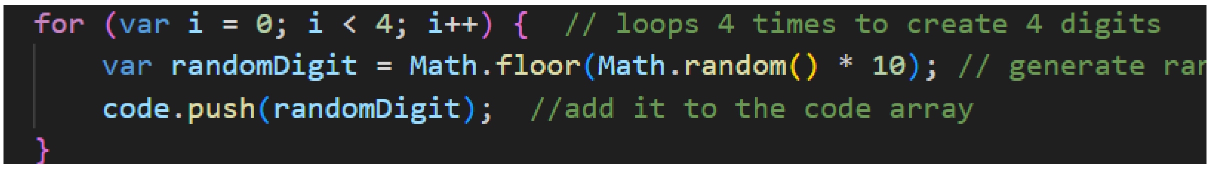
SecretCode – stores the 4 digit secret code

CurrentGuess – stores the players current input

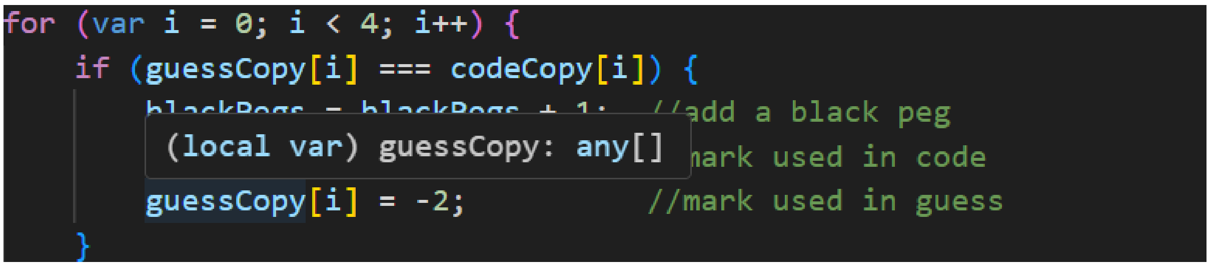
Attempts – stores all previous attempts

HighScore- stores high scores

Loop example:



```
for (var i = 0; i < 4; i++) {  // loops 4 times to create 4 digits
    var randomDigit = Math.floor(Math.random() * 10); // generate ran
    code.push(randomDigit);  //add it to the code array
}
```

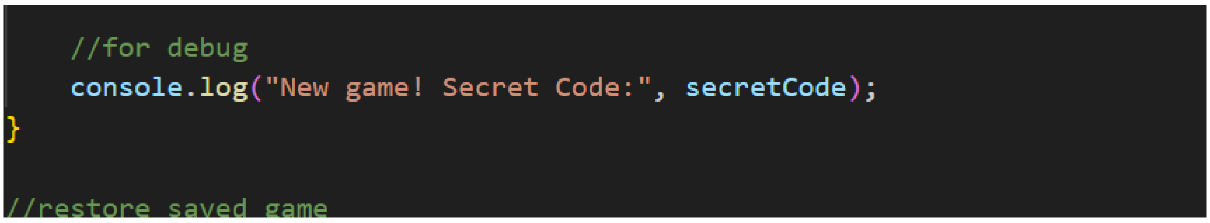


```
for (var i = 0; i < 4; i++) {
    if (guessCopy[i] === codeCopy[i]) {
        blackPegs = blackPegs + 1;  //add a black peg
        (local var) guessCopy: any[] mark used in code
        guessCopy[i] = -2;          //mark used in guess
    }
```

8. Testing and Debugging

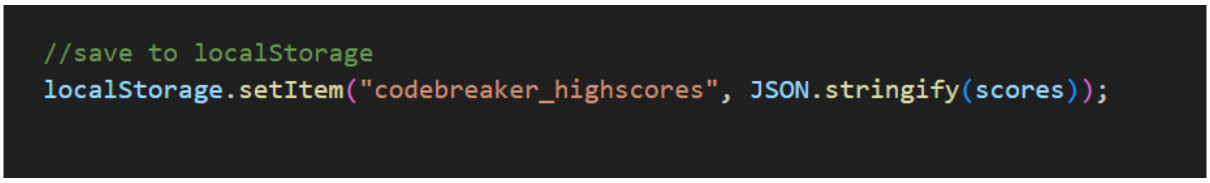
Testing Methods Used:

1. **Console..log Statements:** Used with development to track variable values and functions:



```
    //for debug
    console.log("New game! Secret Code:", secretCode);
}

//restore saved game
```



```
  //save to localStorage
  localStorage.setItem("codebreaker_highscores", JSON.stringify(scores));
```

2. **Browser Develop Tools**

- Chrome tools for debugging
- Elements tab to inspect DOM
- Application tab to view localStorage data

3. **Manual Testing**

Played multiple complete games to test multiple scenarios

Debugging Process:

Bug 1: Double pegs

Problem: same digit counted for both times.

Solution: marked matched digits with special values (-1,2) to prevent it.

Code fix: Added marking system in checkGuess function.

Bug 2: localStorage errors

Problem: JSON.parse failed on empty storage.

Solution: Added null check .

Code fix: var stored= localStorage.getItem(“key”);

If (sorted) {return JSON.parse(stored)} return [];

Bug 3: Array comparison

Problem: array1 === array2 always returned false.

Solution: Compared individual elements with using loop.

9. Technologies Used

- HTML
- CSS
- JavaScript

- Browser Compatibility:
- Chrome
- Mozilla
- MS Edge
- Safari

10. Known Limitations

- It doesn't have any multiplayer support.
- High scores will stored per browser not across devices.
- Game only works browsers with JavaScript enabled.
- LocalStorage requires browser permissions.

------------END OF DOCUMENT--------------