Name: Onkar Anil Waghmode

Div: TY-CS-D

Roll No.: 81

PRN: 12210334

Assignment 2: Design and develop a website to demonstrate (a) breadth first and depth first search for integer elements using JavaScript.

index.html

```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
  <title>BFS & DFS Demo</title>
  <link rel="stylesheet" href="style.css" />
</head>
<body>
  <h1>BFS & DFS Traversal</h1>
  <div class="controls">
    <input type="number" id="nodeValue" placeholder="Enter integer" />
    <button onclick="insertNode()">Insert</button>
    <button onclick="performBFS()">BFS</button>
    <button onclick="performDFS()">DFS</button>
  </div>

  <div id="treeContainer"></div>
  <div id="output"></div>

  <script src="script.js"></script>
</body>
</html>
```

style.css

```
body {
    font-family: Arial, sans-serif;
    text-align: center;
    padding: 20px;
  }

  .controls {
    margin-bottom: 20px;
  }

  input, button {
    padding: 8px 12px;
    margin: 5px;
  }

  #treeContainer {
    display: flex;
    justify-content: center;
    flex-wrap: wrap;
    margin-top: 20px;
  }

  .node {
    margin: 5px;
    padding: 10px;
    border: 2px solid #333;
    border-radius: 50%;
    width: 40px;
    height: 40px;
    line-height: 20px;
    display: inline-block;
    background-color: lightgray;
    transition: background 0.5s;
  }

  .visited {
    background-color: lightgreen !important;
  }

  #output {
    margin-top: 30px;
    font-size: 18px;
  }

  .tree {
    display: flex;
    justify-content: center;
    align-items: flex-start;
    flex-direction: column;
  }

  .level {
    display: flex;
    justify-content: center;
    margin: 20px 0;
  }

  .node {
    width: 40px;
    height: 40px;
    background-color: lightgray;
    border: 2px solid #333;
    border-radius: 50%;
    display: flex;
    justify-content: center;
    align-items: center;
    margin: 0 10px;
    position: relative;
    transition: background 0.5s;
  }

  .visited {
    background-color: lightgreen !important;
  }

  .connector {
    height: 20px;
    width: 2px;
    background-color: black;
    position: absolute;
    top: -20px;
    left: 50%;
    transform: translateX(-50%);
  }
```

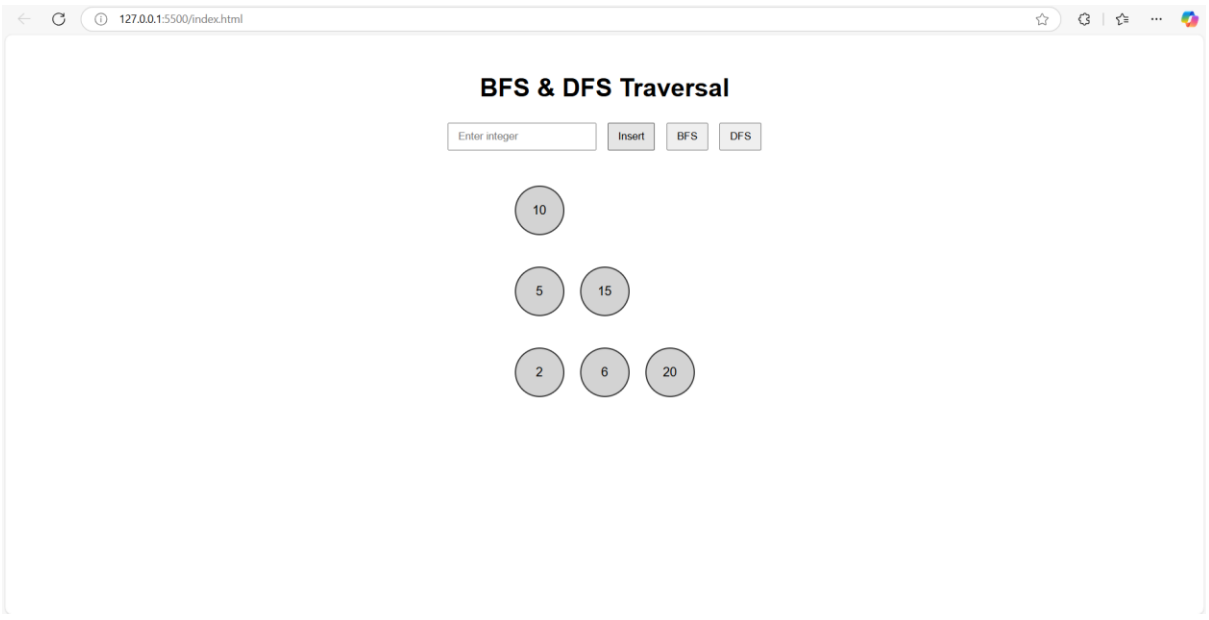
script.js

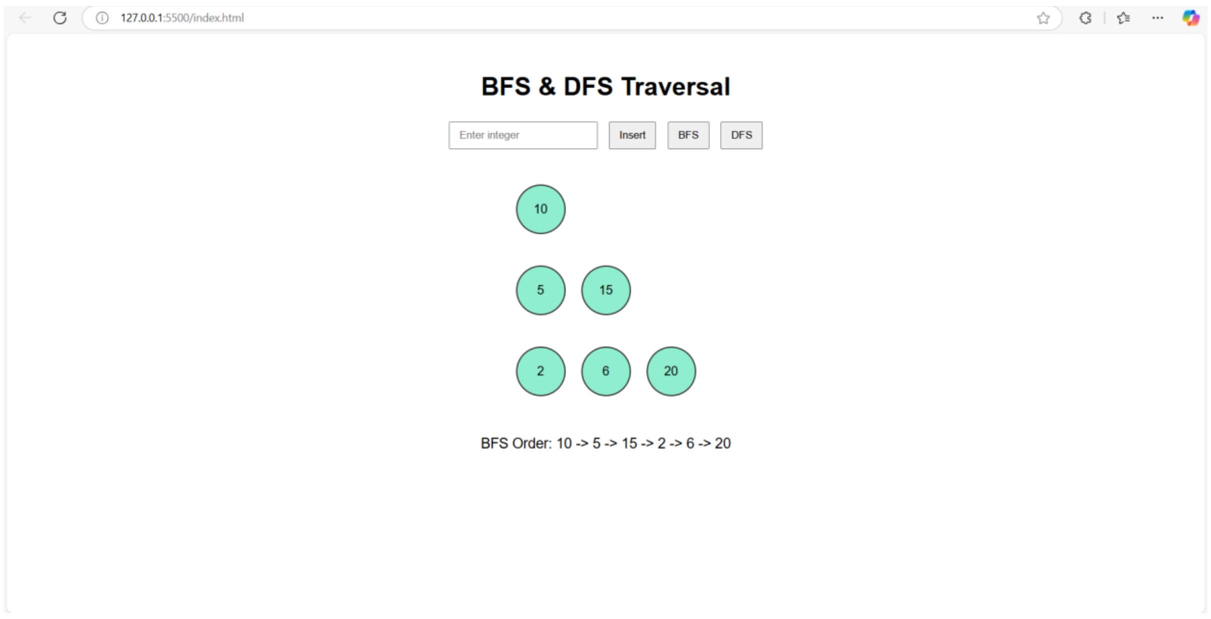
```
class TreeNode {
    constructor(value) {
```

```
    this.value = value;
    this.left = null;
    this.right = null;
  }
}

let root = null;

function insertNode() {
  const value = parseInt(document.getElementById('nodeValue').value);
  if (!isNaN(value)) {
    root = insertIntoTree(root, value);
    renderTree(root);
    document.getElementById('nodeValue').value = '';
    document.getElementById('output').innerText = '';
  }
}

function insertIntoTree(node, value) {
  if (!node) return new TreeNode(value);
  if (value < node.value) node.left = insertIntoTree(node.left, value);
  else node.right = insertIntoTree(node.right, value);
  return node;
}

function renderTree(root) {
  const container = document.getElementById('treeContainer');
  container.innerHTML = '';
  if (!root) return;

  const treeDiv = document.createElement('div');
  treeDiv.className = 'tree';

  const levels = [];

  const queue = [{ node: root, level: 0 }];
  while (queue.length > 0) {
    const { node, level } = queue.shift();

    if (!levels[level]) {
      const levelDiv = document.createElement('div');
      levelDiv.className = 'level';
      levels[level] = levelDiv;
      treeDiv.appendChild(levelDiv);
    }

    const nodeDiv = document.createElement('div');
    nodeDiv.className = 'node';
```

```
      nodeDiv.id = `node-${node.value}`;
      nodeDiv.innerText = node.value;

      levels[level].appendChild(nodeDiv);

      if (node.left) queue.push({ node: node.left, level: level + 1 });
      if (node.right) queue.push({ node: node.right, level: level + 1 });
    }

    container.appendChild(treeDiv);
  }


  function performBFS() {
    if (!root) return;
    const queue = [root];
    const visitedOrder = [];
    resetNodeColors();
    let delay = 0;

    while (queue.length) {
      const node = queue.shift();
      visitedOrder.push(node.value);
      highlightNode(node.value, delay);
      delay += 500;
      if (node.left) queue.push(node.left);
      if (node.right) queue.push(node.right);
    }

    setTimeout(() => {
      document.getElementById('output').innerText = `BFS Order:
${visitedOrder.join(' -> ')}`;
    }, delay);
  }

  function performDFS() {
    const visitedOrder = [];
    resetNodeColors();
    let delay = 0;

    function dfs(node) {
      if (!node) return;
      visitedOrder.push(node.value);
      highlightNode(node.value, delay);
      delay += 500;
      dfs(node.left);
      dfs(node.right);
    }
    dfs(root);

    setTimeout(() => {
      document.getElementById('output').innerText = `DFS Order:
${visitedOrder.join(' -> ')}`;
    }, delay);
  }

  function highlightNode(value, delay) {
    setTimeout(() => {
      const nodeDiv = document.getElementById(`node-${value}`);
      if (nodeDiv) {
        nodeDiv.classList.add('visited');
      }
    }, delay);
  }

  function resetNodeColors() {
    document.querySelectorAll('.node').forEach((el) => {
      el.classList.remove('visited');
    });
  }
```

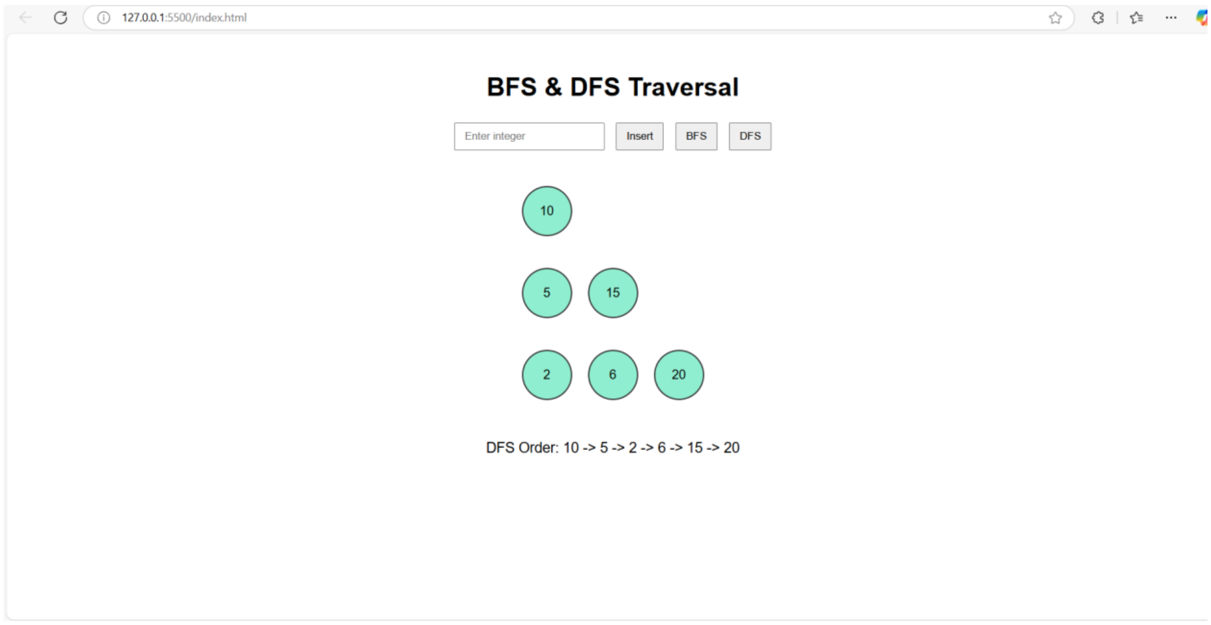
Output:





BFS & DFS Traversal

BFS Order: 10 -> 5 -> 15 -> 2 -> 6 -> 20



BFS & DFS Traversal

DFS Order: 10 -> 5 -> 2 -> 6 -> 15 -> 20