Instructions:

A user can provide input to this programming by clicking on the HTML buttons that have the text content “Rock,” “Paper,” or “Sicssor.” These three inputs are really the only inputs that the user has to worry about; by selecting either of these “attack modes,” the game has already started.

HTML CODE

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Rock Paper Scissors</title>

  <!--custom script and style imports-->
  <script src="index.js" defer></script>
  <link rel="stylesheet" href="./index.css" />

  <!--Bootstrap script and style imports-->
  <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/
bootstrap.min.css" integrity="sha384-ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/
iJTQUOhcWr7x9JvoRxT2MZw1T" crossorigin="anonymous">
  <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js" integrity="sha384-q8i/
X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo"
crossorigin="anonymous"></script>
  <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper.min.js"
integrity="sha384-
UO2eT0CpHqdSJQ6hJty5KVphtPhzWj9WO1clHTMGa3JDZwrnQq4sF86dIHNDz0W1"
crossorigin="anonymous"></script>
  <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js"
integrity="sha384-JjSmVgyd0p3pXB1rRibZUAYoIIy6OrQ6VrjIEaFf/nJGzIxFDsf4x0xIM+B07jRM"
crossorigin="anonymous"></script>

</head>
<body>
  <h3 id="main-title">Play Rock, Paper, Scissor Against The Computer!</h3>
  <h5 id="winner-header"></h5>
  <div id="cta-btns-wrapper">
    <button id="new-game-btn" class="btn btn-success" disabled="true">New Game</button>
    <button id="reset-scores-btn" class="btn btn-danger" disabled="true">Reset Scores</
button>
    <button type="button" class="btn btn-primary view-player-log-btn" disabled="true" data-
toggle="modal" data-target="#exampleModal1">
      View Your Scores
    </button>
```

```
      <button type="button" class="btn btn-primary view-computer-log-btn" disabled="true"
data-toggle="modal" data-target="#exampleModal2">
        View Computer Scores
      </button>
    </div>
    <main id="game-area">
      <section id="player-attack-options-wrapper">
        <ul id="player-attack-options">
          <div>
            <h4>Choose Your Attack</h4>
          </div>
          <div class="attack-btn-wrapper">
            <button id="rock-btn" class="attack-btn btn btn-info">Rock</button>
          </div>
          <div class="attack-btn-wrapper">
            <button id="paper-btn" class="attack-btn btn btn-info">Paper</button>
          </div>
          <div class="attack-btn-wrapper">
            <button id="scissor-btn" class="attack-btn btn btn-info">Scissor</button>
          </div>
        </ul>
      </section>
      <section id="attacks-wrapper">
        <ul id="attacks">
          <div>
            <h4>Attacks</h4>
          </div>
          <h5>You Selected: </h5>
          <span id="player-attack-icon"></span>
          <h5>Computer Selected:</h5>
          <span id="computer-attack-icon"></span>
        </ul>
      </section>
      <section id="scores-wrapper">
        <ul id="scores">
          <div>
            <h4>Scores</h4>
          </div>
          <h5>Your Score: </h5>
          <span id="player-score">0</span>
          <h5>Computer Score:</h5>
          <span id="computer-score">0</span>
        </ul>
      </section>
    </main>
   <div class="modal fade" id="exampleModal1" tabindex="-1" role="dialog" aria-
labelledby="exampleModalLabel" aria-hidden="true">
    <div class="modal-dialog" role="document">
      <div class="modal-content">
        <div class="modal-header">
```

```
        <h5 class="modal-title" id="exampleModalLabel">Player Game Log</h5>
        <button type="button" class="close" data-dismiss="modal" aria-label="Close">
          <span aria-hidden="true">&times;</span>
        </button>
      </div>
      <div class="player-modal-body">

      </div>
      <div class="modal-footer">
        <button type="button" class="btn btn-secondary" data-dismiss="modal">Close</button>
      </div>
    </div>
  </div>
 </div>
 <div class="modal fade" id="exampleModal2" tabindex="-1" role="dialog" aria-
labelledby="exampleModalLabel" aria-hidden="true">
  <div class="modal-dialog" role="document">
    <div class="modal-content">
      <div class="modal-header">
        <h5 class="modal-title" id="exampleModalLabel">Computer Game Log</h5>
        <button type="button" class="close" data-dismiss="modal" aria-label="Close">
          <span aria-hidden="true">&times;</span>
        </button>
      </div>
      <div class="computer-modal-body">

      </div>
      <div class="modal-footer">
        <button type="button" class="btn btn-secondary" data-dismiss="modal">Close</button>
      </div>
    </div>
  </div>
 </div>
</body>
</html>
```

CSS CODE

```
@import url("https://fonts.googleapis.com/css2?family=Montserrat&display=swap");

* {
 box-sizing: border-box;
 font-family: "Montserrat", sans-serif;
}

#game-area {
 width: fit-content;
 margin: auto;
 box-shadow: rgba(0, 0, 0, 0.35) 0px 5px 15px;
 display: flex;
 align-items: center;
}

#player-attack-options-wrapper,
#attacks-wrapper,
#scores-wrapper {
 padding: 10px;
 border-right: 1px solid rgba(128, 128, 128, 0.658);
 width: fit-content;
}

#main-title {
 margin-top: 5%;
 text-align: center;
 margin-bottom: 5%;
 letter-spacing: 1px;
}

#winner-header {
 text-align: center;
 height: 30px;
}

#cta-btns-wrapper {
 display: flex;
 align-items: center;
 justify-content: center;
 margin: 20px auto 20px auto;
}
```

```
.single-game-wrapper {
 margin: 20px;
 text-align: left;
 border: 1px solid black;
 padding: 0px;
}

.single-game-wrapper li {
 margin: 5px;
 list-style: none;
 text-align: left;
}

#cta-btns-wrapper button {
 margin: 10px;
}

#player-attack-options,
#attacks,
#scores {
 display: flex;
 flex-direction: column;
 align-items: center;
 padding: 0px;
}

#scores-wrapper,
.attack-btn {
 border: none;
}

#attacks,
#scores {
 display: flex;
 align-items: center;
 flex-direction: column;
}

#player-attack-icon,
#computer-attack-icon,
#player-score,
#computer-score {
 height: 40px;
}

.single-game-log-wrapper {
 border: 1px solid grey;
 margin: 20px;
}
```

```
.single-game-log {
 padding: 0px;
}

.modal-body {
 max-height: 500px;
 overflow: scroll;
}

.single-game-log li {
 list-style: none;
 margin: 5px;
}

.attack-btn-wrapper {
 list-style: none;
 margin: 15px;
 font-size: 1.5rem;
}

.game-log-key {
 font-size: 10000px;
}

.attack-btn {
 width: 120px;
 padding: 10px;
 border-radius: 0.25rem;
}

#attacks h5,
#scores h5 {
 margin: 20px;
}

h4 {
 font-weight: bold !important;
 border-bottom: 1px solid grey;
}

button[disabled] {
 cursor: not-allowed;
}
```

TYPESCRIPT CODE

```
// The following icons were taken from https://icons8.com/
const rockIcon =
'<img src="https://img.icons8.com/emoji/20/000000/rock-emoji.png"/>';
const paperIcon =
'<img src="https://img.icons8.com/officel/20/000000/paper.png"/>';
const scissorIcon =
'<img src="https://img.icons8.com/ios/20/000000/sewing-scissors.png"/>';

// Global HTML element variables that will be accessed across multiple functions
const newGameBtn = document.getElementById("new-game-btn");
const resetScoresBtn = document.getElementById("reset-scores-btn");
const playerModal = document.querySelector('.player-modal-body');
const computerModal = document.querySelector('.computer-modal-body');
const viewPlayerGameLogBtn = document.querySelector(".view-player-log-btn");
const viewComputerGameLogBtn = document.querySelector(".view-computer-log-btn");
```

```
const attackBtns = document.querySelectorAll(".attack-btn"); // Include Rock, Paper, and
Scissor buttons

const winnerHeader = document.getElementById('winner-header');

const playerAttackIcon = document.getElementById("player-attack-icon");
const computerAttackIcon = document.getElementById("computer-attack-icon");

const playerScore = document.getElementById("player-score");
const computerScore = document.getElementById("computer-score");

// This reusable function disables the Rock, Paper, and Scissor buttons, making them unable to
be clicked
function disablePlayerAttackButtons(disableState: boolean): void {
 const attackBtnRock = document.getElementById("rock-btn");
 const attackBtnPaper = document.getElementById("paper-btn");
 const attackBtnScissor = document.getElementById("scissor-btn");

 disableBtn(attackBtnRock, disableState);
 disableBtn(attackBtnPaper, disableState);
 disableBtn(attackBtnScissor, disableState);
}

// Code snipptes like this: const {functionName} = (args) =>
// Are anonymous arrow functions that do not require a return statement nor the "function"
keyword

// Renders the HTML element button passed as the first argument, unclickable
const disableBtn = (element: Element, disableState: boolean) => (element.disabled =
disableState);

// Sets the HTML value for the element passed as the first argument
const setInnerHTML = (element: Element, value: string) => element.innerHTML = value;

// Sets the text content for the HTML element passed as the first argument
const setTextContent = (element: HTMLElement, text: any) => (element.textContent = text);

// Sets the icon of the selected attack method on the HTML element passed as the first argument
const setAttackIcon = (element: HTMLElement, icon: string, iconDescription: string) =>
 (element.innerHTML = `${icon} (${iconDescription})`);

// These two variables will store the player and computer attacks, whether it be "Rock", "Paper",
or "Scissor"
let playerAttack: string;
let computerAttack: string;

const playerGameLog = []; // Array that stores the result of games played by player
const computerGameLog = []; // Array that stores the result of games played by computer

// Alternative to a for-loop
```

```
attackBtns.forEach((btn): void => {
 // Adding a click listener for each button
 btn.addEventListener("click", (): void => {
   disablePlayerAttackButtons(true);

   showPlayerAttackIcon(btn);
   showComputerAttackIcon();
   calculateGameWinner();

   disableBtn(newGameBtn, false);
   disableBtn(viewPlayerGameLogBtn, false);
   disableBtn(viewComputerGameLogBtn, false);
   disableBtn(resetScoresBtn, false);
 });
});

function showPlayerAttackIcon(btn: Element) {
 const btnAttackType = btn.id.split("-btn")[0]; // Returns the attack type of the HTML button
clicked

 if (btnAttackType === "rock") {
   setAttackIcon(playerAttackIcon, rockIcon, "Rock");
   playerAttack = "Rock";
 } else if (btnAttackType === "paper") {
   setAttackIcon(playerAttackIcon, paperIcon, "Paper");
   playerAttack = "Paper";
 } else {
   setAttackIcon(playerAttackIcon, scissorIcon, "Scissor");
   playerAttack = "Scissor";
 }
}

function showComputerAttackIcon(): void {
 const randomComputerAttackValue = Math.floor(Math.random() * 3); // Random value from 0 to
2, inclusive

 if (randomComputerAttackValue === 0) {
   setAttackIcon(computerAttackIcon, rockIcon, "Rock");
   computerAttack = "Rock";
 } else if (randomComputerAttackValue === 1) {
   setAttackIcon(computerAttackIcon, paperIcon, "Paper");
   computerAttack = "Paper";
 } else {
   setAttackIcon(computerAttackIcon, scissorIcon, "Scissor");
   computerAttack = "Scissor";
 }
}

function setWinnerHeaderTextContent(text: string) {
 winnerHeader.textContent = text;
```

```
}

function calculateGameWinner() {
 if (playerAttack === "Rock" && computerAttack === "Rock") {
   setScore(false, "Rock", "Rock");
   setWinnerHeaderTextContent('TIE! No one wins!');
 } else if (playerAttack === "Rock" && computerAttack === "Paper") {
   setScore("computer", "Rock", "Paper");
   setWinnerHeaderTextContent("YOU LOSE! Paper beats Rock!");
 } else if (playerAttack === "Rock" && computerAttack === "Scissor") {
   setScore("player", "Rock", "Scissor");
   setWinnerHeaderTextContent('YOU WIN! Rock beats Scissor!');
 } else if (playerAttack === "Paper" && computerAttack === "Rock") {
   setScore("player", "Paper", "Rock");
   setWinnerHeaderTextContent('YOU LOSE! Rock beats Paper!');
 } else if (playerAttack === "Paper" && computerAttack === "Paper") {
   setScore(false, "Paper", "Paper");
   setWinnerHeaderTextContent('TIE! No one wins!');
 } else if (playerAttack === "Paper" && computerAttack === "Scissor") {
   setScore("computer", "Paper", "Scissor");
   setWinnerHeaderTextContent('YOU LOSE! Scissor beats Paper!');
 } else if (playerAttack === "Scissor" && computerAttack === "Rock") {
   setScore("computer", "Scissor", "Rock");
   setWinnerHeaderTextContent('YOU LOSE! Rock beats Scissor!');
 } else if (playerAttack === "Scissor" && computerAttack === "Paper") {
   setScore("player", "Scissor", "Paper");
   setWinnerHeaderTextContent('YOU LOSE! Scissor beats Paper!');
 } else if (playerAttack === "Scissor" && computerAttack === "Scissor") {
   setScore(false, "Scissor", "Scissor");
   setWinnerHeaderTextContent('TIE! No one wins!');
 }
}

function setScore(winner: string | boolean, playerSelection: string, computerSelection: string) {
 let result: string;

 if (!winner) {
   result = "TIE!";
 } else if (winner === "player") {
   console.log('player won')
   setTextContent(playerScore, parseInt(playerScore.textContent) + 1);
   result = "PLAYER WON";
 } else {
   console.log('computer won')
   setTextContent(computerScore, parseInt(computerScore.textContent) + 1);
   result = "COMPUTER WON!";
 }

 // Everything to the left of the colon is a key; everything to the right is the value
 // Pushing an object with 2 key-value pairs into the playerGameLog list
```

```
 playerGameLog.push({
   selection: playerSelection,
   result: result
 });

 computerGameLog.push({
   selection: computerSelection,
   result: result,
 })
}

viewComputerGameLogBtn.addEventListener("click", (): void => {
 viewGameLog(false);
});

viewPlayerGameLogBtn.addEventListener("click", (): void => {
 viewGameLog(true);
});

function viewGameLog(retrievePlayerScores: boolean) {

 let gameLogArray: Array<object>;
 let modal: Element;

 setInnerHTML(playerModal, null);
 setInnerHTML(computerModal, null);

 if (!retrievePlayerScores) {
   modal = computerModal;
   gameLogArray = [...computerGameLog];
 } else {
   modal = playerModal;
   gameLogArray = [...playerGameLog];
 }

 // Each object within gameLogArray represents the result of one game played
 for (let i = 0; i < gameLogArray.length; i++) {

   const singleGameWrapper = document.createElement('ul'); // creating an unordered list
   const selection = document.createElement('li'); // creating a list item
   const result = document.createElement('li');
   const gameNumber = document.createElement('li');

   selection.textContent = `ATTACK TYPE: ${gameLogArray[i].selection}`;
   result.textContent = `GAME RESULT: ${gameLogArray[i].result}`;
   gameNumber.textContent = `GAME #: ${(i + 1).toString()}`;

   singleGameWrapper.appendChild(selection);
   singleGameWrapper.appendChild(result);
   singleGameWrapper.appendChild(gameNumber);
```

```
   singleGameWrapper.className = 'single-game-wrapper';

   modal.appendChild(singleGameWrapper);
  }
 }


// Resets game screen
newGameBtn.addEventListener("click", (): void => {
 disableBtn(newGameBtn, true);
 disablePlayerAttackButtons(false);
 resetIcons();

 setWinnerHeaderTextContent(null);
});

resetScoresBtn.addEventListener("click", (): void => {
 resetIcons();

 setInnerHTML(playerModal, null);
 setInnerHTML(computerModal, null);

 setWinnerHeaderTextContent(null);

 setTextContent(playerScore, "0");
 setTextContent(computerScore, "0");

 disableBtn(viewComputerGameLogBtn, true);
 disableBtn(viewPlayerGameLogBtn, true);
 disableBtn(resetScoresBtn, true);

 playerGameLog.length = 0;
 computerGameLog.length = 0;
});

function resetIcons() {
 setInnerHTML(playerAttackIcon, null);
 setInnerHTML(computerAttackIcon, null);
}
```