Exercises: Sorting Algorithm Visualization

Exercise 1 - Consider the JavaScript Code

```
// Waits until the page is fully loaded before generating bars
document.addEventListener("DOMContentLoaded", () => {
  generateBars();
});

// Function to generate random bars
function generateBars() {
  const barsContainer = document.getElementById("bars");
  barsContainer.innerHTML = ""; // Clears existing bars

  // Generates an array of 10 random heights between 10 and 100 pixels
  let numbers = Array.from(
    { length: 10 },
    () => Math.floor(Math.random() * 100) + 10
  );

  numbers.forEach((num) => {
    let bar = document.createElement("div");
    bar.classList.add("bar");
    bar.style.height = `${num}px`;
    barsContainer.appendChild(bar);
  });
}
```

```
// Function to disable sorting buttons during sorting
function disableButtons() {
  document.querySelector("button[onclick='bubbleSort()']").disabled = true;
  document.querySelector("button[onclick='quickSort()']").disabled = true;
}

// Function to enable sorting buttons
function enableButtons() {
  document.querySelector("button[onclick='bubbleSort()']").disabled = false;
  document.querySelector("button[onclick='quickSort()']").disabled = false;
}

// Function to perform the bubble sort algorithm on the bars
async function bubbleSort() {
  disableButtons();
  // TODO: Implement the code for the Bubble Sort algorithm

  enableButtons();
}

// Function to perform the quick sort algorithm on the bars
async function quickSort(low = 0, high = null) {
  let bars = document.querySelectorAll(".bar");

  if (high === null) high = bars.length - 1;
```

```
  if (low < high) {
    let pivotIndex = await partition(bars, low, high);
    await quickSort(low, pivotIndex - 1);
    await quickSort(pivotIndex + 1, high);
  }

  // Enable buttons after sorting is completely done (only when recursion ends)
  if (low === 0 && high === bars.length - 1) {
    enableButtons();
  }
}

// Partition function for Quick Sort
async function partition(bars, low, high) {
  let pivot = parseInt(bars[high].style.height);
  let i = low - 1;

  for (let j = low; j < high; j++) {
    let heightJ = parseInt(bars[j].style.height);

    if (heightJ < pivot) {
      i++;
      await swap(bars[i], bars[j]);
    }
  }

  await swap(bars[i + 1], bars[high]);
```

```
  return i + 1;
}

// Function to swap two bars (divs)
function swap(bar1, bar2) {
  return new Promise((resolve) => {
    setTimeout(() => {
      let temp = bar1.style.height;
      bar1.style.height = bar2.style.height;
      bar2.style.height = temp;
      resolve();
    }, 200);
  });
}
```

Hint:

These are the correct lines of code put in a random order.

```
let len = bars.length; // Number of bars

await swap(bars[j], bars[j + 1]); // Calls the swap function (with a delay for visualization)

let height1 = parseInt(bars[j].style.height); // Gets the height of the first bar

// Inner loop iterates through unsorted part of the array
for (let j = 0; j < len - 1 - i; j++) {
```

```
  // What goes inside here?
}

// If the current bar is taller than the next one, swap them
if (height1 > height2) {
  // What goes inside here?
}

let height2 = parseInt(bars[j + 1].style.height); // Gets the height of the next bar

// Outer loop iterates through the array
for (let i = 0; i < len - 1; i++) {
  // What goes inside here?
}

let bars = document.querySelectorAll(".bar"); // Select all bars
```