JavaScript Assignment - 06

Name – Soumitra Anil Kode
Roll no. – 42134 , BE- 06, Batch - P6

DOP :

Git Repo Link :
Source Code :

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8" />
 <meta name="viewport" content="width=device-width, initial-scale=1" />
 <title>Array Operations - Creative Edition</title>
 <style>
  @import
url('https://fonts.googleapis.com/css2?family=Work+Sans:wght@400;600&display=s
wap');
  body {
   background: linear-gradient(135deg, #f0f4f8 0%, #d9e2f3 100%);
   font-family: 'Work Sans', Arial, sans-serif;
   margin: 0;
   padding: 36px 18px;
   display: flex;
   justify-content: center;
   min-height: 100vh;
   align-items: flex-start;
   color: #224166;
   user-select: none;
  }
  .container {
   width: 420px;
   background: #fcfdffcc;
   border-radius: 18px;
   padding: 30px 26px 40px 26px;
   box-shadow: 0 10px 28px #a7bacddd;
   backdrop-filter: saturate(180%) blur(14px);
   border: 1.8px solid #c7d9f3;
   animation: fadeSlideIn 0.7s ease forwards;
  }
  @keyframes fadeSlideIn {
   from {opacity: 0; transform: translateY(30px);}
   to {opacity: 1; transform: translateY(0);}
  }
  h1 {
   font-weight: 700;
   font-size: 1.58em;
```

```
  margin-bottom: 14px;
  text-align: center;
  color: #1c2d4a;
  letter-spacing: 0.045em;
  user-select: text;
}
h2 {
  font-weight: 650;
  font-size: 1.22em;
  color: #2f466b;
  border-bottom: 2px solid #7ea4d2;
  padding-bottom: 6px;
  margin-bottom: 22px;
  user-select: text;
}
label {
  font-weight: 600;
  font-size: 1.02em;
  color: #28384e;
  margin-top: 18px;
  display: inline-block;
  user-select: text;
}
input[type="number"] {
  width: 100%;
  padding: 12px 15px;
  font-size: 1.15em;
  border: 2px solid #aac1dc;
  border-radius: 9px;
  margin-top: 6px;
  font-weight: 500;
  transition: border-color 0.3s;
  color: #1e2a47;
}
input[type="number"]:focus {
  outline: none;
  border-color: #4575e6;
  background-color: #e7efff;
}
button {
  margin-top: 12px;
  font-weight: 700;
  font-size: 1.06em;
  color: #f9faff;
  background: linear-gradient(90deg, #3778f0 10%, #4a63d9 95%);
  border: none;
  border-radius: 12px;
```

```
  padding: 11px 16px;
  box-shadow: 0 3px 19px #4a63d947;
  cursor: pointer;
  transition: background 0.24s ease, box-shadow 0.3s ease;
  user-select: none;
  width: 100%;
}
button:hover, button:focus {
  background: linear-gradient(90deg, #526bdb 10%, #5366cf 90%);
  box-shadow: 0 6px 22px #525bfd5c;
  outline: none;
  transform: scale(1.03);
}
.btn-row {
  display: flex;
  gap: 14px;
  margin-top: 12px;
}
.btn-row button {
  width: 50%;
  padding: 14px 0;
  font-weight: 700;
}
.section {
  margin-top: 28px;
  user-select: text;
}
.array-display {
  background: #e6f0fd;
  border: 1.5px solid #9abcffaa;
  border-radius: 14px;
  padding: 13px 18px;
  font-family: 'Courier New', monospace;
  font-size: 1.12em;
  color: #2350a0;
  min-height: 35px;
  margin-top: 9px;
  overflow-x: auto;
  white-space: nowrap;
  word-wrap: normal;
}
#message, #searchMessage {
  margin-top: 20px;
  font-weight: 700;
  font-size: 1.07em;
  color: #2d4f87;
  min-height: 26px;
```

```
    user-select:none;
    text-align: center;
    transition: color 0.27s ease;
   }
  </style>
</head>
<body>
  <div class="container" role="main" aria-label="Array Operations Interface">
   <h1>Array Operations</h1>

   <!-- Array Size -->
   <div>
    <label for="arraySize">Enter size of array:</label>
    <input type="number" id="arraySize" min="1" aria-describedby="sizeHelp" aria-
label="Array size input">
    <button onclick="setSize()" aria-live="polite">Set Size</button>
   </div>

   <!-- Add Elements -->
   <div id="addElementBox" class="section" style="display:none;">
    <label for="elementInput">Enter element:</label>
    <input type="number" id="elementInput" aria-label="Element input">
    <button onclick="addElement()" aria-live="polite">Add Element</button>
    <p id="sizeMessage" style="margin-top:5px; color:#647a99;" aria-
live="polite"></p>
   </div>

   <!-- Current Array Display -->
   <div class="section" aria-live="polite" aria-atomic="true">
    <strong>INPUT ARRAY FOR OPERATIONS:</strong>
    <div id="currentArray" class="array-display">[]</div>
   </div>

   <!-- Remove Element Section -->
   <div class="section">
    <h2>Remove Element</h2>
    <label for="removeValue">Enter value to remove:</label>
    <input type="number" id="removeValue" aria-label="Value to remove">
    <div class="btn-row">
     <button onclick="removeNormal()" aria-live="polite">Remove (Normal
method)</button>
     <button onclick="removeSplice()" aria-live="polite">Remove (Using
splice)</button>
    </div>
    <strong>Array After Remove Operation:</strong>
    <div id="removeArray" class="array-display">[]</div>
   </div>
```

```
  <!-- Search Section -->
  <div class="section">
    <h2>Search for Element</h2>
    <label for="searchValue">Enter value to search:</label>
    <input type="number" id="searchValue" aria-label="Value to search">
    <div class="btn-row">
      <button onclick="searchLinear()" aria-live="polite">Search (Linear
Search)</button>
      <button onclick="searchIncludes()" aria-live="polite">Search (Using
includes)</button>
    </div>
    <div id="searchMessage" aria-live="polite" style="min-height: 30px; font-weight:
600; color: #395682;"></div>
  </div>

  <!-- Empty Array Section -->
  <div class="section">
    <h2>Empty the Array</h2>
    <div class="btn-row">
      <button onclick="emptyNormal()" aria-live="polite">Empty (Set to [])</button>
      <button onclick="emptySplice()" aria-live="polite">Empty (Using
splice)</button>
    </div>
    <strong>Array After Empty Operation:</strong>
    <div id="emptyArray" class="array-display">[]</div>
  </div>

  <p id="message" aria-live="polite" style="margin-top:22px;"></p>
 </div>

 <script>
  let arr = [];
  let maxSize = 0;

  function setSize() {
    maxSize = parseInt(document.getElementById("arraySize").value);
    if (isNaN(maxSize) || maxSize <= 0) {
      showMessage("Please enter a valid array size.");
      return;
    }
    arr = [];
    displayMasterArray();
    clearOperationDisplays();
    clearSearchMessage();
    showMessage(`Array size set to ${maxSize}. Add elements one by one.`);
    document.getElementById("addElementBox").style.display = "block";
```

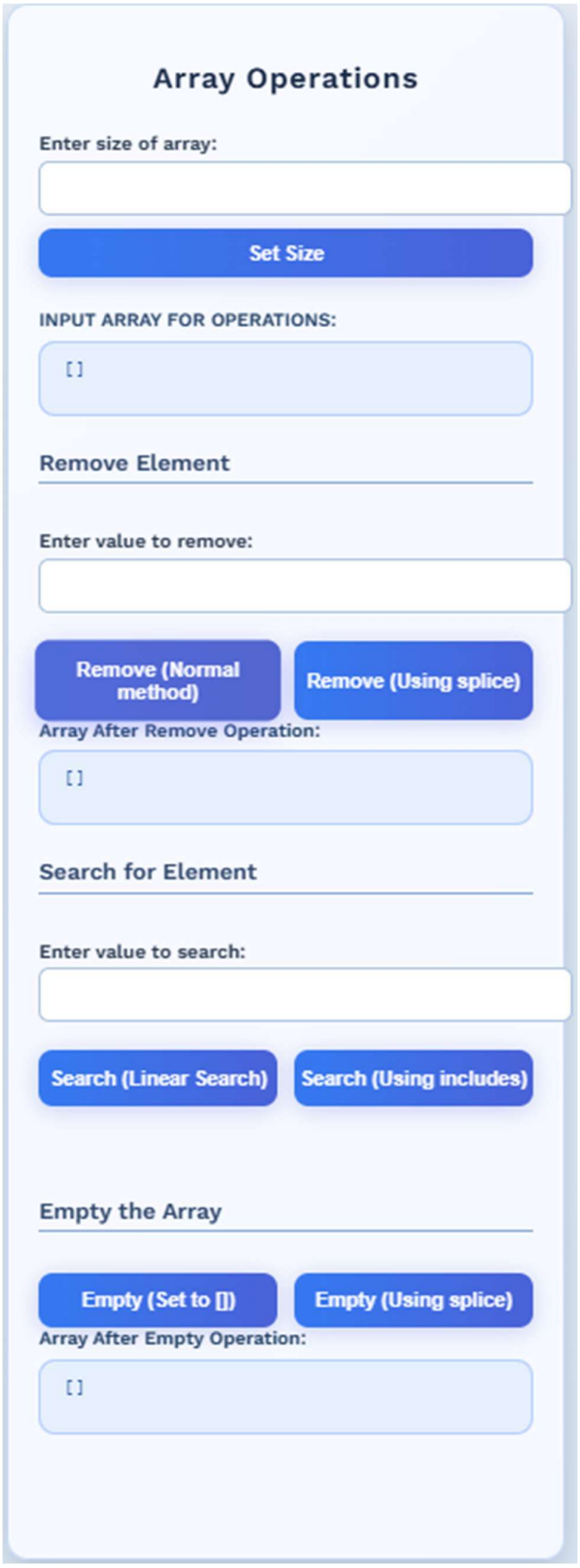
```
    document.getElementById("sizeMessage").textContent = `0 / ${maxSize}
elements added.`;
   }

   function addElement() {
    if (arr.length >= maxSize) {
     showMessage("Array is already full.");
     return;
    }
    const val = parseInt(document.getElementById("elementInput").value);
    if (isNaN(val)) {
     showMessage("Please enter a valid integer.");
     return;
    }
    arr.push(val);
    document.getElementById("elementInput").value = "";
    displayMasterArray();
    document.getElementById("sizeMessage").textContent = `${arr.length} /
${maxSize} elements added.`;
    if (arr.length === maxSize) {
     showMessage("Array creation completed.");
    }
   }

   function removeNormal() {
    const val = parseInt(document.getElementById("removeValue").value);
    if (isNaN(val)) {
     showMessage("Please enter a valid number to remove.");
     return;
    }
    const filteredArr = arr.filter(item => item !== val);
    displayMasterArray();
    displayRemoveArray(filteredArr);
    if (filteredArr.length < arr.length) {
     showMessage("Value removed using normal method.");
    } else {
     showMessage("Value not found.");
    }
   }

   function removeSplice() {
    const val = parseInt(document.getElementById("removeValue").value);
    if (isNaN(val)) {
     showMessage("Please enter a valid number to remove.");
     return;
    }
    let newArr = arr.slice();
```

```
  const index = newArr.indexOf(val);
  if (index !== -1) {
    newArr.splice(index, 1);
    displayMasterArray();
    displayRemoveArray(newArr);
    showMessage("Value removed using splice.");
  } else {
    displayRemoveArray(newArr);
    showMessage("Value not found.");
  }
}

function searchLinear() {
  const val = parseInt(document.getElementById("searchValue").value);
  if (isNaN(val)) {
    showSearchMessage("Please enter a valid number to search.");
    return;
  }
  let foundIndex = -1;
  for (let i = 0; i < arr.length; i++) {
    if (arr[i] === val) {
      foundIndex = i;
      break;
    }
  }
  if (foundIndex !== -1) {
    showSearchMessage(`Element found at index: ${foundIndex}`);
  } else {
    showSearchMessage("Element not found.");
  }
}

function searchIncludes() {
  const val = parseInt(document.getElementById("searchValue").value);
  if (isNaN(val)) {
    showSearchMessage("Please enter a valid number to search.");
    return;
  }
  const index = arr.indexOf(val);
  if (index !== -1) {
    showSearchMessage(`Element found at index: ${index}`);
  } else {
    showSearchMessage("Element not found.");
  }
}

function emptyNormal() {
```

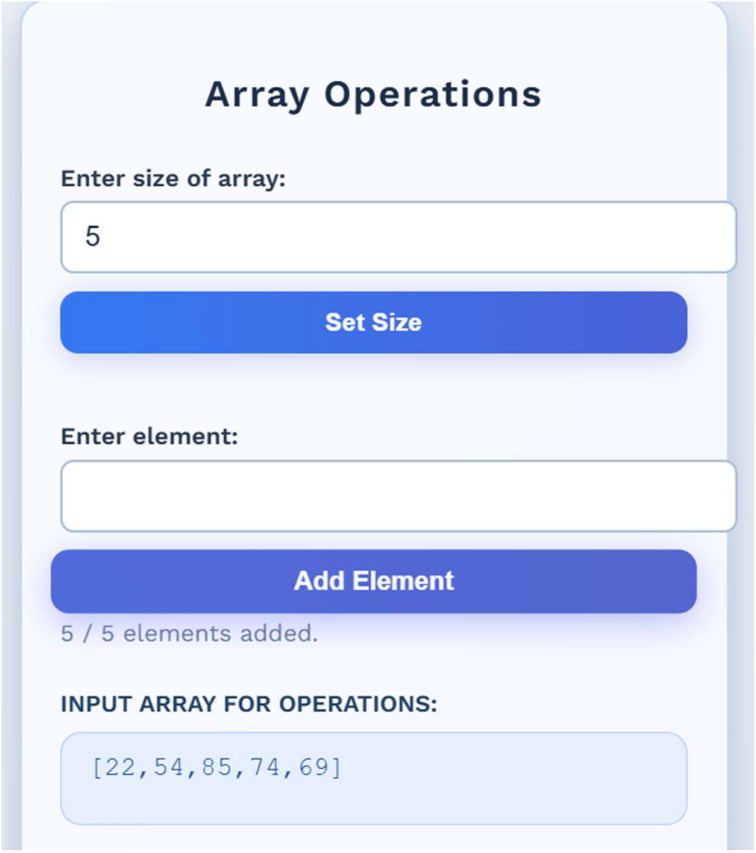
```
    arr = [];
    displayMasterArray();
    displayEmptyArray(arr);
    showMessage("Array emptied using normal method.");
   }

   function emptySplice() {
    arr.splice(0, arr.length);
    displayMasterArray();
    displayEmptyArray(arr);
    showMessage("Array emptied using splice().");
   }

   function displayMasterArray() {
    document.getElementById("currentArray").textContent = JSON.stringify(arr);
   }
   function displayRemoveArray(arrayToShow) {
    document.getElementById("removeArray").textContent =
JSON.stringify(arrayToShow);
   }
   function displayEmptyArray(arrayToShow) {
    document.getElementById("emptyArray").textContent =
JSON.stringify(arrayToShow);
   }
   function clearOperationDisplays() {
    document.getElementById("removeArray").textContent = "[]";
    document.getElementById("emptyArray").textContent = "[]";
   }
   function showMessage(msg) {
    document.getElementById("message").textContent = msg;
   }
   function showSearchMessage(msg) {
    document.getElementById("searchMessage").textContent = msg;
   }
   function clearSearchMessage() {
    document.getElementById("searchMessage").textContent = "";
   }
  </script>
</body>
</html>
```

Output –

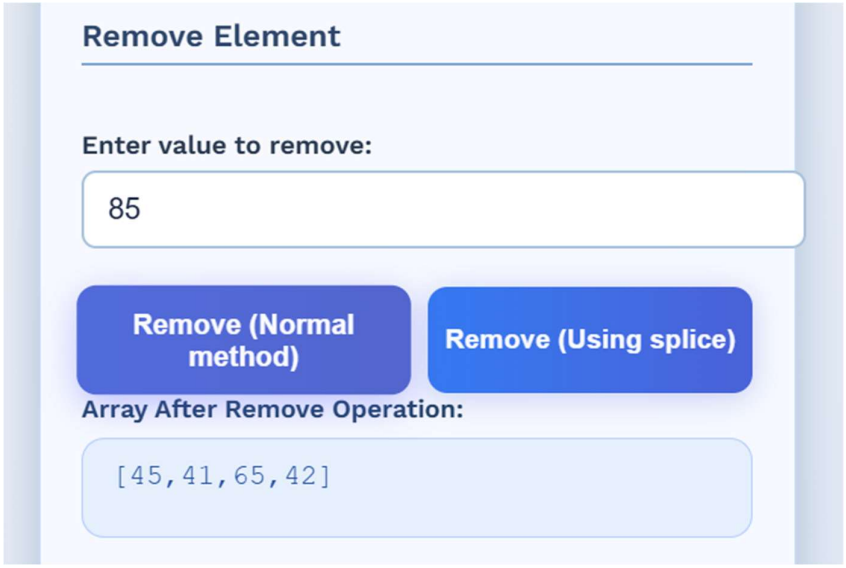
A. Initial Landing Page –



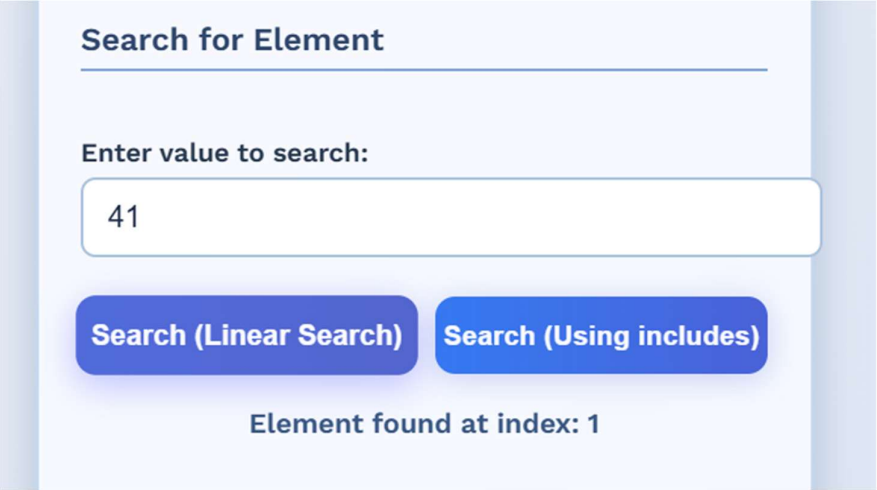
B. Output after initializing array –



C. Deleting an element from array –



D. Output after searching an element in array –



E. Output after emptying the entire array –