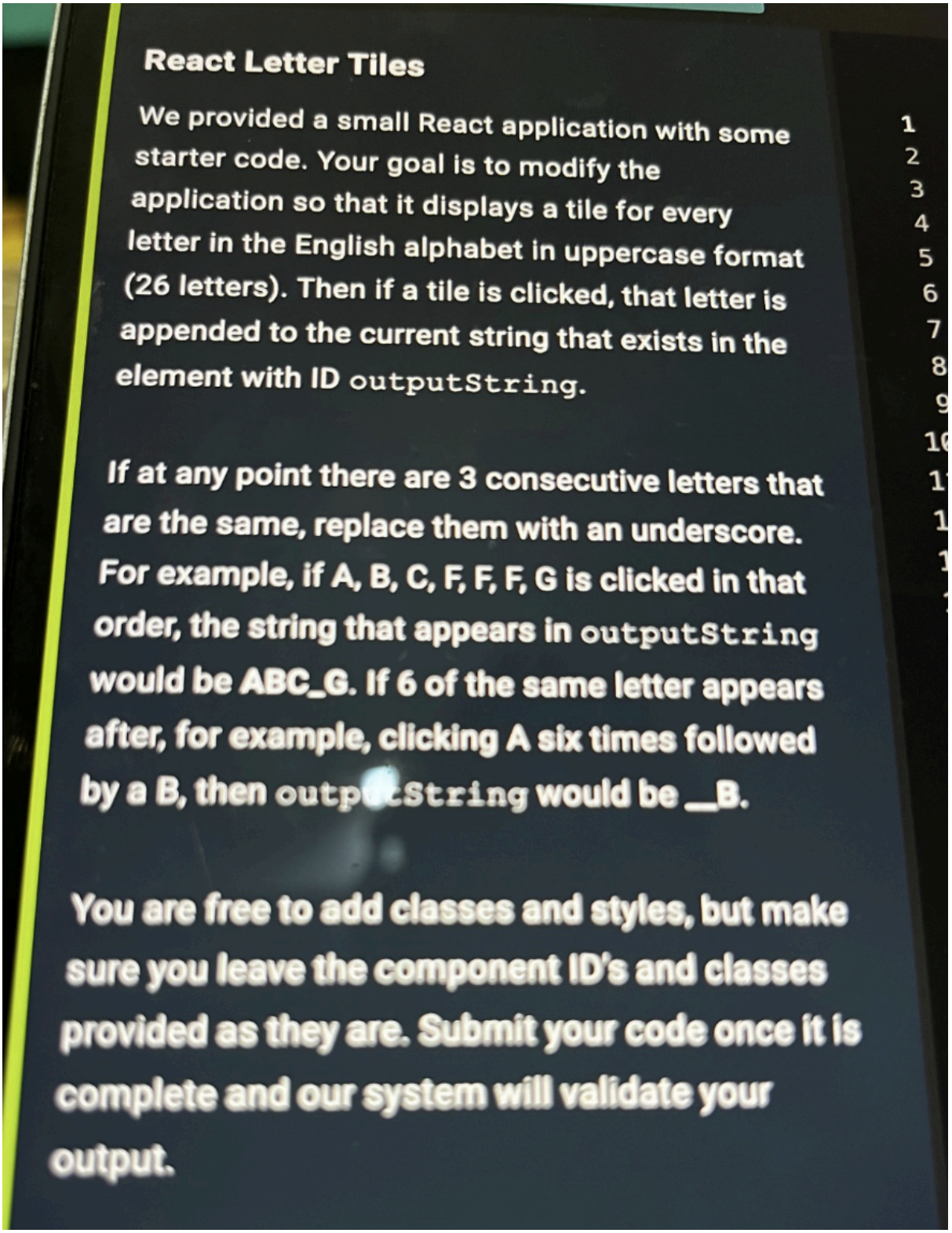
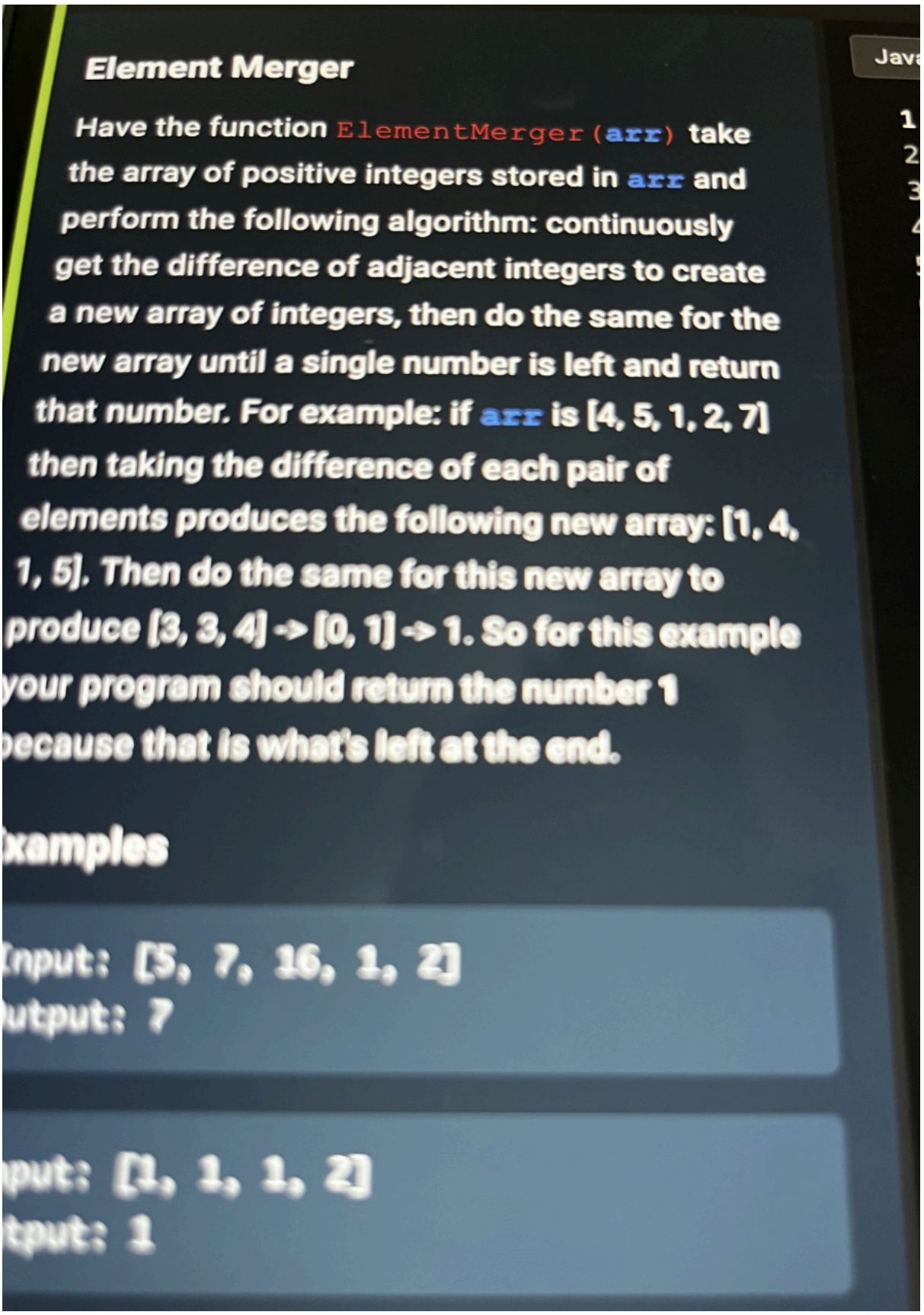
Round 1 DS/ALGO




```
import React, { useState } from 'react';
import { createRoot } from 'react-dom/client';
```

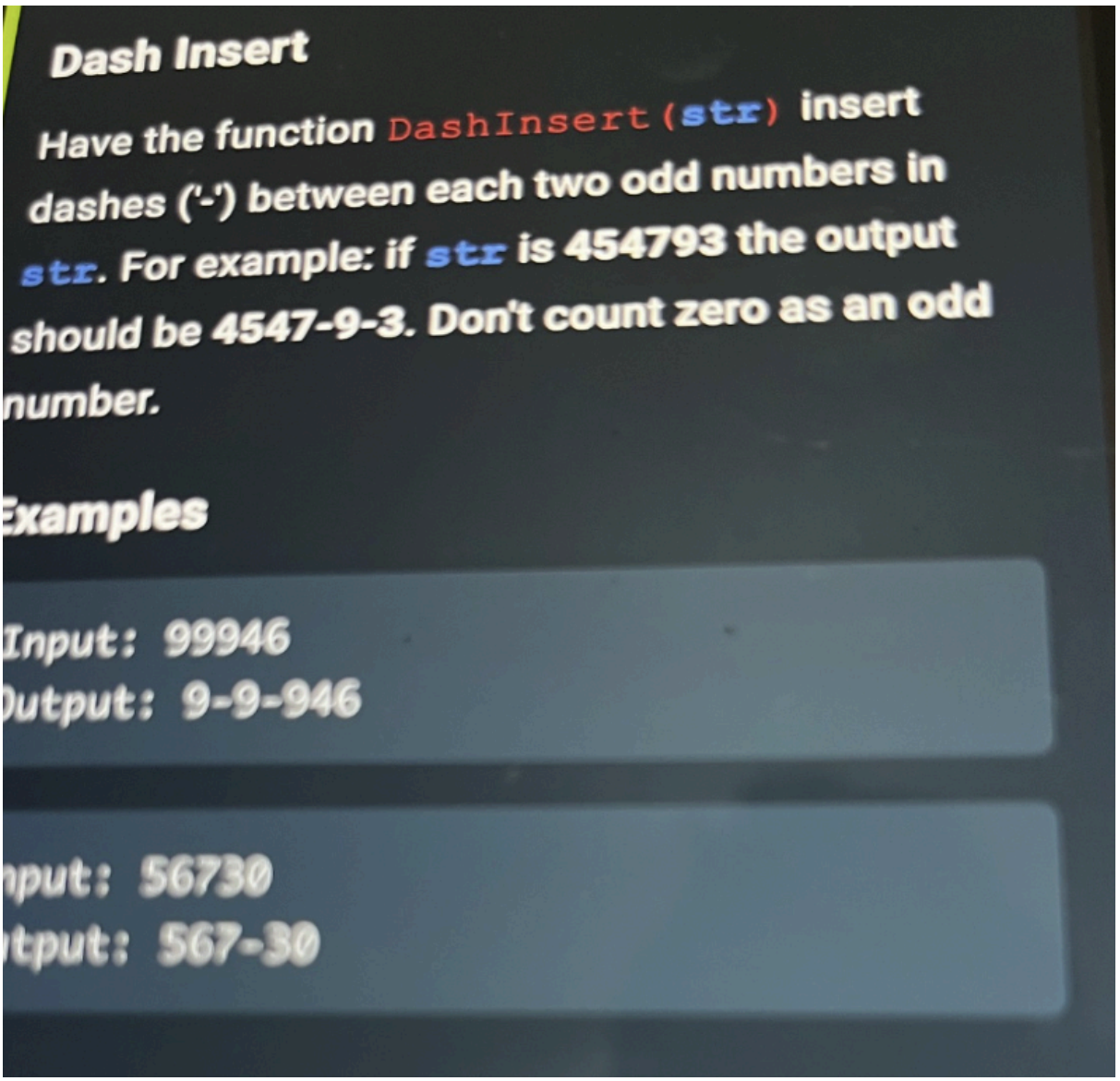
```
const style = {
  letterContainer: {
    overflow: 'auto',
    marginBottom: '10px'
  },
  letter: {
    float: 'left',
    padding: '10px 10px',
    background: '#c9e4ed',
    borderRadius: '5px',
    marginRight: '5px',
    cursor: 'pointer'
  }
};

function Tile({ letter, onClick }) {
  return (
    <button style={style.letter} onClick={() => onClick(letter)}>
      {letter}
    </button>
  );
}

function Application() {
  const [output, setOutput] = useState("");

  const handleClick = (newChar) => {
    const newString = output + newChar;
    const processed = processString(newString);
    setOutput(processed);
  };

  // Function to replace 3+ same consecutive characters with '_'
  const processString = (input) => {
    let result = "";
    let i = 0;

    while (i < input.length) {
      let j = i;
      while (j < input.length && input[j] === input[i]) {
        j++;
      }

      // Count of repeated characters = j - i
      if (j - i >= 3) {
        result += "_"; // Replace repeated block with underscore
      } else {
        result += input.slice(i, j); // Keep original part
```

```
      }

      i = j;
    }

    return result;
  };

  const alphabet = Array.from({ length: 26 }, (_, i) =>
    String.fromCharCode(65 + i)
  ); // ['A', 'B', ..., 'Z']

  return (
    <section>
      <aside style={style.letterContainer} id="letterContainer">
        {alphabet.map((letter) => (
          <Tile key={letter} letter={letter} onClick={handleClick} />
        ))}
      </aside>
      <div id="outputString">{output}</div>
    </section>
  );
}

const root = createRoot(document.getElementById('root'));
root.render(<Application />);
```




```
function ElementMerger(arr) {
  // Repeat until the array has only 1 element
```

```
  while (arr.length > 1) {
    const newArr = [];

    // For each adjacent pair, push the absolute difference
    for (let i = 0; i < arr.length - 1; i++) {
      newArr.push(Math.abs(arr[i] - arr[i + 1]));
    }

    // Replace the old array with the new difference array
    arr = newArr;
  }

  // Return the only remaining element
  return arr[0];
}

// Example usage:
console.log(ElementMerger([4, 5, 1, 2, 7])); // Output: 1
```




```
function DashInsert(str) {
  let result = "";

  for (let i = 0; i < str.length; i++) {
    const current = str[i];
    const next = str[i + 1];

    result += current;

    // Check if both current and next digits are odd (and next exists)
    if (next && isOdd(current) && isOdd(next)) {
      result += "-"; // Insert dash
    }
  }

  return result;
```

```
}

// Helper to check if digit is odd
function isOdd(ch) {
  const num = parseInt(ch);
  return num % 2 === 1;
}

// keep this function call here
console.log(DashInsert(readline()));
```