

Round 1 DS/ALGO

React Letter Tiles

We provided a small React application with some starter code. Your goal is to modify the application so that it displays a tile for every letter in the English alphabet in uppercase format (26 letters). Then if a tile is clicked, that letter is appended to the current string that exists in the element with ID `outputString`.

If at any point there are 3 consecutive letters that are the same, replace them with an underscore. For example, if A, B, C, F, F, F, G is clicked in that order, the string that appears in `outputString` would be `ABC_G`. If 6 of the same letter appears after, for example, clicking A six times followed by a B, then `outputString` would be `_B`.

You are free to add classes and styles, but make sure you leave the component ID's and classes provided as they are. Submit your code once it is complete and our system will validate your output.

```
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 />);

```


Element Merger

Have the function `ElementMerger(arr)` take the array of positive integers stored in `arr` and perform the following algorithm: continuously get the difference of adjacent integers to create a new array of integers, then do the same for the new array until a single number is left and return that number. For example: if `arr` is `[4, 5, 1, 2, 7]` then taking the difference of each pair of elements produces the following new array: `[1, 4, 1, 5]`. Then do the same for this new array to produce `[3, 3, 4] -> [0, 1] -> 1`. So for this example your program should return the number 1 because that is what's left at the end.

Examples

Input: `[5, 7, 16, 1, 2]`

Output: 7

Input: `[1, 1, 1, 2]`

Output: 1

```
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
```

Dash Insert

Have the function `DashInsert(str)` insert dashes ('-') between each two odd numbers in `str`. For example: if `str` is `454793` the output should be `4547-9-3`. Don't count zero as an odd number.

Examples

Input: 99946

Output: 9-9-946

Input: 56730

Output: 567-30

```
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()));
```