Generating All Subsequences of a String

```
//The substring() method extracts characters, between two indices (positions),
//from a string, and returns the substring.
// The substring() method extracts characters from start to end (exclusive).
// The substring() method does not change the original string.

function stringSubsequence(input, output){
    if(input.length === 0){
        console.log(output)
        return
    }
    stringSubsequence(input.substring(1), output);
    stringSubsequence(input.substring(1), output + input[0]);
}

const inputString = "abc";
stringSubsequence(inputString, "");

// printSubsequences("123", "")
// ├─ printSubsequences("23", "")
// │  ├─ printSubsequences("3", "")
// │  │  ├─ printSubsequences("", "")      // outputs ""
// │  │  └─ printSubsequences("", "3")     // outputs "3"
// │  └─ printSubsequences("3", "2")
// │     ├─ printSubsequences("", "2")     // outputs "2"
// │     └─ printSubsequences("", "23")    // outputs "23"
// └─ printSubsequences("23", "1")
//    ├─ printSubsequences("3", "1")
//    │  ├─ printSubsequences("", "1")     // outputs "1"
//    │  └─ printSubsequences("", "13")    // outputs "13"
//    └─ printSubsequences("3", "12")
//       ├─ printSubsequences("", "12")
//       └─ printSubsequences("", "123")   // outputs "123"
```

Union of Two Arrays

```
const array1 = [1, 2, 3];
const array2 = [2, 3, 4];

function unionArraySize(a,b){
    const combinedArray = a.concat(b);
    const unionSet = new Set(combinedArray)

    return {size: unionSet.size, set:unionSet };
}

const result = unionArraySize(array1, array2);
console.log(result)
```

Intersection of Two Sorted Arrays

```
const array1 = [1, 2, 5,3];
const array2 = [2, 3, 4];

function quickSort(arr){
    if (arr.length < 2) {
        return arr;
      }
    let pivot = arr[arr.length - 1]
    let left = []
    let right = []
    for(let i = 0; i < arr.length - 1; i++) {
        if(arr[i] < pivot) {
            left.push(arr[i]);
        } else {
            right.push(arr[i]);
        }
    }
    return [...quickSort(left), pivot, ...quickSort(right)];
}

const sortedArray1 = quickSort(array1);
const sortedArray2 = quickSort(array2);

function intersections(arr1, arr2){
    let i = 0;
    let j = 0;
    let intersect = [];
    while(i < arr1.length && j < arr2.length){
        if(arr1[i] === arr2[j]){
            intersect.push(arr1[i]);
            i++;
            j++;
        } else if(arr1[i] > arr2[j]){
            j++;
        } else if(arr1[i] < arr2[j]){
            i++;
        }
    }
    return intersect;
}

console.log(intersections(sortedArray1,sortedArray2))
```