Developer Written Screening Questions

Note:
This technical screening document is not an example of the quality of code you will find in our team. It is an example of code specific to the screening process to determine your technical and problem solving skills as well as attention to detail. We use this as a base assessment to determine that your skills match what we are looking for in our Software Developers before moving to the phone and in person interviews.

We are looking for you to understand the algorithms and be able to provide test cases that ensure accuracy of the implementation and answers to questions that show a solid understanding of the code.

Speedy Pizza is a pizza chain with dozens of busy restaurants in a densly populated region. Their marketing department has come up with what they think is a brilliant promotion: the One in a Million Challenge. Any customer that orders a three topping pizza with a unique combination of toppings across all pizzas in all stores for that month will receive a coupon for a free pizza by email (if they provide their email address).

Each month they generate a file containing a JSON representation of all the pizzas ordered that month. The toppings for each pizza are sorted in alphabetical order in this form:

```
[{"email": "email1@example.com", "toppings":
["Mushrooms","Pepperoni","Peppers"]},
{"email": "email2@example.com", "toppings":
["Cheddar","Garlic","Oregano"]},
{"email": "email3@example.com", "toppings": ["Bacon","Ham","Pineapple"]},
{"email": "", "toppings": ["Parmesan","Tomatoes"]},
{"email": "email4@example.com", "toppings":
["Mushrooms","Pepperoni","Peppers"]},
{"email": "", "toppings": ["Cheddar","Tomatoes"]},
{"email": "email5@example.com", "toppings": ["Bacon","Ham","Pineapple"]},
{"email": "email6@example.com", "toppings": ["Beef","Parmesan"]},
{"email": "", "toppings": ["Onions","Pepperoni"]},
{"email": "", "toppings": ["Bacon","Ham","Pineapple"]}]
```

Speedy Pizza wants a routine to load this data and print out the email addresses of any winners. (For the above example, it would print out email2@example.com.) Two algorithms are suggested, represented by these implementations:

```
function printWinners1(inputArray) {
    // Perform a QuickSort on the array.  We provide an anonymous function
to do the comparisons.
    inputArray.sort((a,b)=>{
        // Convert each toppings array to a string and do a string
comparison
        return a.toppings.toString().localeCompare(b.toppings.toString());
    });

    let previousEmail = '';
    let previousToppingsAsString = '';
    let previousToppingCount = 0;
    let numberOfSimilarOrders = 0;
    // Iterate through the array, with "order" being each item in the
array.
    inputArray.map((order) => {
        let toppingsAsString = order.toppings.toString();
        if (toppingsAsString === previousToppingsAsString) {
            numberOfSimilarOrders++;
        } else {
            if ((numberOfSimilarOrders === 1) && (previousToppingCount ===
3) && (previousEmail) !== '') {
                // Print out the email.
                console.log(previousEmail);
            }
            previousToppingsAsString = toppingsAsString;
            previousEmail = order.email;
            previousToppingCount = order.toppings.length;
            numberOfSimilarOrders = 1;
        }
    });
}
```

```
function printWinners2(inputArray) {
    let hashTable = new Map();

    // Iterate through the array, with "order" being each item in the
array.
    inputArray.map((order) => {
        if ((order.toppings.length === 3) && (order.email !== '')){
            let toppingsAsString = order.toppings.toString();

            let matchingValue = hashTable.get(toppingsAsString);
            if (matchingValue) {
                // This key was already in the hash table.
                // matchingValue is a reference to the object in the hash
table.
                matchingValue.duplicate = true;
            } else {
                // Insert into the hash table, using the toppings as the
key and an object containing the email as the value.
                hashTable.set(toppingsAsString,{email: order.email,
duplicate: false});
            }
        }
    });

    // Iterate through the values in the hash table, with "value" being
each value.
    hashTable.forEach((value) => {
        if (!value.duplicate) {
            // Print out the email.
            console.log(value.email);
        }
    });
}
```

Questions

1. Describe some data sets that you would use to test any proposed implementation and its accuracy.
2. Did you identify any bugs in these proposed implementations?
3. Once any small bugs are fixed, will either of these approaches give us the output we're looking for?
4. What are some advantages or disadvantages of these approaches?
5. If we were querying a database instead of working with an in-memory structure, would we use a significantly different algorithm to find the winners? If so, how would you expect that algorithm to differ in performance and why?