Q1: There is an array, each item has such format:
{firstName: 'xxx', lastName: 'xxx', customerID: 'xxx', note: 'xxx', profession: ‘xxx’},
lastName, note can be empty, customerID can only be a set of digital numbers, profession can only have ‘student’, ‘freelancer’, ‘productOwner’, ’engineer’ or ‘systemAnalytics’.

(1) Please follow the principle (‘firstName’ + ‘lastName’ + ‘customerID’) to sort this array and print it out. function sortUserName(user) {}

(2) Please sort by ‘profession’ to follow the principle.
(‘systemAnalytics’ > ‘engineer’ > ‘productOwner‘> ‘freelancer‘ > ‘student‘)
function sortByType(user) {}

A:

```
function sortUserName(user) {
    function customSort(a, b) {
        let A = (a.firstName + a.lastName + a.customerID).toLowerCase();
        let B = (b.firstName + b.lastName + b.customerID).toLowerCase();
        if (A < B) return -1;
        if (A > B) return 1;
        return 0;
    }
    const arr1 = user.sort(customSort);
    console.log(arr1);
}
function sortByType(user) {
    function customSort(a, b) {
        const proOrder = {
            'systemAnalytics': 1,
            'engineer': 2,
            'productOwner': 3,
            'freelancer': 4,
            'student': 5
        };
        const proA = proOrder[a.profession];
        const proB = proOrder[b.profession];
        return proA - proB;
    }
    const arr2 = user.sort(customSort);
    console.log(arr2);
```

```
}
const user = [
    { firstName: 'John', lastName: 'Doe', customerID: '123', note: '★★★', profession:
'student' },
    { firstName: 'Tom', lastName: '', customerID: '133', note: '★', profession:
'engineer' },
    { firstName: 'Joan', lastName: '', customerID: '009', note: '★★★★★', profession:
'productOwner' },
    { firstName: 'Jane', lastName: 'Smith', customerID: '001', note: '★★★★',
profession: 'freelancer' },
    { firstName: 'Alice', lastName: '', customerID: '000', note: '★★', profession:
'systemAnalytics' }
];
console.log("---Sort By User Name---")
sortUserName(user);
let copyArray = [...user];
console.log("---Sort By Profession---")
sortByType(copyArray);
```

Q2: Explain why does this color not works, and how to fix make it work on 1st list? Write styling make every other line give background color to next one:

A:

Because the class name for <li> under both <ol> and <ul> tags is 'item', all elements with '.item' class will be set to blue. Therefore, items in both lists will be affected. Problem can be solved by setting the text color of the 1st 'shop-list' of the two classes 'shop-list' to blue using ':nth-child(1)'. The following is the fixed code:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <style>
        .container {
            font-size: 14px;
        }
        .container .header {
            font-size: 18px;
        }
```

```
        .container .shop-list {
            list-style: none;
            margin-left: -15px;
        }
        .container .shop-list li.item {
            color: green;
        }
        .container .shop-list:nth-child(1) .item {
            color: blue;
        }
        .container .shop-list li:nth-child(even) {
            background-color: lightpink;
        }
        .container .shop-list li:nth-child(odd) {
            background-color: lightblue;
        }
    </style>
</head>
<body>
    <div class="container">
        <div class="header">5/8  外出確認表</div>
        <div class="content">
            <ol class="shop-list">
                <li class="item">麵包</li>
                <li class="item">短袖衣服</li>
                <li class="item">飲用水</li>
                <li class="item">帳篷</li>
            </ol>
            <ul class="shop-list">
                <li class="item">暈車藥</li>
                <li class="item">感冒藥</li>
                <li class="item">丹木斯</li>
                <li class="item">咳嗽糖漿</li>
            </ul>
        </div>
        <div class="footer">以上僅共參考</div>
    </div>
</body>
</html>
```

Q3: let items = [1, 1, 1, 5, 2, 3, 4, 3, 3, 3, 3, 3, 3, 7, 8, 5, 4, 9, 0, 1, 3, 2, 6, 7, 5, 4, 4, 7, 8, 8, 0, 1, 2, 3, 1]; Please write down a function to console log unique value from this array. function getUniqueNumber (items) { }

A:

```
function getUniqueNumber(items) {
  console.log(new Set(items));
}
let items = [1, 1, 1, 5, 2, 3, 4, 3, 3, 3, 3, 3, 3, 7, 8, 5, 4, 9, 0, 1, 3, 2, 6, 7, 5, 4, 4, 7, 8, 8, 0, 1, 2, 3, 1];
getUniqueNumber(items);
```

Q4: Can you explain about Interface and Enum, and where will you be using, please make some examples.

A:

Where to Use:

Interface: Used for defining the structure of objects.

Enum: Used for defining a set of named constants.

When to Use:

Use an interface when you want to define the shape of an object.

Use an enum when you want to define a set of related named constants.

Define an interface for a clothes:

```
interface Clothes {
  style: string;
  color: string;
  piece: number;
}
```

Using the Clothes interface:

```
let myClothes: Clothes = {
  style: "Skirt",
  color: "Red",
  piece: 1000
};
console.log(myClothes.style); // Skirt
console.log(myClothes.color); // Red
console.log(myClothes.piece); // 1000
```

Define an enum for different colors:

```
enum Color {
   Red,
   Blue,
   Pink
}
```

Using the Color enum:

```
let myColor: Color = Color.Pink;
console.log(myColor); //Pink
```

Q5: Can you explain the problem with the following code, and how to fix it.

A:

The problem with the current code is that setState in React is asynchronous, meaning it does not immediately update the state. When you call setState multiple times in a row like in handleAddCount, React batches the state updates together for performance reasons. Therefore, when you call this.setState({ count: this.state.count + 1 }) three times in a row, React will combine them into a single state update, resulting in count being incremented only once.

To fix this, you can use the functional form of setState that takes the previous state as an argument. This ensures that each update is based on the previous state and will correctly increment count three times.

Here's the corrected code:

```
handleAddCount() {
   this.setState(prevState => ({ count: prevState.count + 1 }));
   this.setState(prevState => ({ count: prevState.count + 1 }));
   this.setState(prevState => ({ count: prevState.count + 1 }));
}
```

Q6: Please write the sample code to debounce handleOnChange.

A:

```
import React, { useState } from 'react';
import debounce from 'lodash.debounce';

function SearchBox() {
   const [value, setValue] = useState('');
   function handleOnChange(event) {
      const debounced = debounce(() => {
```

```
            console.log(event.target.value);//make Ajax call
        }, 5000); //延遲 5 秒執行
        debounced();
        setValue(event.target.value);
    }
    return <input type="search" name="p" value={value} onChange={handleOnChange}
/>;
}
export default SearchBox;
```