```
** All following exams please using Javascript only **
1.
/**
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’.
**/
/**
Q1. Please follow the principle (‘firstName’ + ‘lastName’ + ‘customerID’)
to sort this array and print it out.
**/

/* ANSWER: */

function sortUserName(user) {
  if (!user) return;

  let userArr = user.slice();
  //   console.log(userArr);

  function compareFn(userA, userB) {
    const getCompareStr = (user) =>
      user.firstanme + user.lastname + String(user.customerId);
    if (getCompareStr(userA) < getCompareStr(userB)) {
      return -1;
    } else if (getCompareStr(userA) > getCompareStr(userB)) {
      return 1;
    }
    return 0;
  }

  console.log(userArr.sort(compareFn));
}

/**
Q2. Please sort by ‘profession’ to follow the principle.
(‘systemAnalytics’ > ‘engineer’ > ‘productOwner’ > ‘freelancer’ >
‘student’’)
**/

/* ANSWER: */

function sortByType(user) {
  if (!user) return;

  let userArr = user.slice();
  //   console.log(userArr);

  const priority = {
    systemAnalytics: 0,
    engineer: 1,
    productOwner: 2,
    freelancer: 3,
    student: 4,
  };

  function compareFn(userA, userB) {
    if (priority[userA.profession] < priority[userB.profession]) {
      return -1;
    } else if (priority[userA.profession] > priority[userB.profession]) {
      return 1;
    }
    return 0;
  }
  console.log(userArr.sort(compareFn));
}

2.
/** HTML
<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>
**/
/** CSS
.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 .item {
 /* Explain why does this color not works, and how to fix make it work on
1st list */
 color: blue;
}
/* Write styling make every other line give background color to next one */
**/

/* ANSWER for Explain why does this color not works */
/*
This rule (.container .shop-list .item {...}) can NOT override the above rule,
becasue the above rule (.container .shop-list li.item {...}) is more specific
*/

/* ANSWER for , and how to fix make it work on 1st list */
/*
.container .shop-list:nth-child(2n + 1) li.item {
    color: blue;
}
*/

/* ANSWER for Write styling make every other line give background color to next
one */
/*
Sorry, I do not know the answer,
Please give me some hint/feedback if you have free time.
*/

3.
/**
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.
**/

/* ANSWER: */

function getUniqueNumber(items) {
  const itemArr = items.slice();

  if (!itemArr) return;

  const uniqueArr = Array.from(new Set(itemArr));
  console.log(uniqueArr);
}

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);

4.
/** Can you explain about Interface and Enum, and where will you be using,
please make some examples. **/

/* ANSWER for interfaces:
An interface defines a contract for the methods or properties that a class or
object should implement,

When I need to create classes that adhere to a certain structure or contract.
I would use interface,

Here's an example:
*/

class PetInterface {
  sound() {
    throw new Error("Method not implemented");
  }
  description() {
    throw new Error("Method not implemented");
  }
}

class Dog extends PetInterface {
  constructor(name) {
    super();
    this.name = name || "Max";
  }

  sound() {
    console.log(`${this.name} barks !!`);
  }

  description() {
    console.log(`${this.name} is a dog`);
  }
}

/* ANSWER for Enum:
Enums are types that contain a limited number of fixed values,

When I assign a limited number of fixed values to a variable, I would use Enum.

Here's an example:
*/

const PetEnum = {
  DOG: Symbol("DOG"),
  CAT: Symbol("CAT"),
  RABBIT: Symbol("RABBIT"),
};

function testEnum(petType) {
  switch (petType) {
    case PetEnum.DOG:
      console.log("The pet is a dog");
      break;
    case PetEnum.CAT:
      console.log("The pet is a cat");
      break;
    case PetEnum.RABBIT:
      console.log("The pet is a rabbit");
      break;
    default:
      console.log("Pet not defined");
  }
}

const pet = { name: "Max", type: PetEnum.RABBIT };
testEnum(pet.type);

5.
/** Can you explain the problem with the following code, and how to fix
it. **/
class Count extends React.Component {
 constructor(props) {
 super(props);
 this.state = { count: 0 };
 this.handleAddCount = this.handleAddCount.bind(this);
 }

 handleAddCount(){
 this.setState({ count: this.state.count + 1 });
 this.setState({ count: this.state.count + 1 });
 this.setState({ count: this.state.count + 1 });
 }
 render() {
 return (
 <div>
 <h2>{this.state.count}</h2>
 <button onClick={this.handleAddCount}>Add</button>
 </div>
 );
 }
}
ReactDOM.render(
 <Count />,
 document.getElementById('root')
);

/* ANSWER */
/*
The problem is about the function handleAddCount()
Becase of rendering algorithm, React batch the three setState() callings to a
SINGLE setSate() calling.

To FIX it, modify the function handleAddCount(),
by callinng setState() based on previous state
*/

handleAddCount() {
  /* FIX it by callinng setState() based on previous state */
  this.setState((prevState) => ({
    ...prevState,
    count: prevState.count + 1,
  }));
  this.setState((prevState) => ({
    ...prevState,
    count: prevState.count + 1,
  }));
  this.setState((prevState) => ({
    ...prevState,
    count: prevState.count + 1,
  }));
}

6.
/** Please write the sample code to debounce handleOnChange **/
var SearchBox = React.createClass({
 render: function() {
 return <input type="search" name="p" onChange={this.handleOnChange}
/>;
 },
 handleOnChange: function(event) {
 // make ajax call
 }
});

/* ANSWER */

export default function SearchBox() {
  const [fetchState, setFetchState] = useState("");
  const [searchQuery, setSearchQuery] = useState("");

  function handleOnChange(event) {
    // make ajax call
    setSearchQuery(event.target.value);
  }

  useEffect(() => {
    const getData = setTimeout(async () => {
      const response = await fetch(
        `https://api.postalpincode.in/pincode/${searchQuery}`
      );
      const parsedData = await response.json();
      setFetchState(parsedData[0].Message);
      console.log(parsedData[0].Message);
    }, 2000);

    return () => clearTimeout(getData);
  }, [searchQuery]);

  return (
    <>
      <p>{`Fetch After 2 sec: ${fetchState}`}</p>
      <input
        type="search"
        name="p"
        onChange={handleOnChange}
        value={searchQuery}
        placeholder="800001"
      />
    </>
  );
}
```