We are creating a simple counter using a useState hook in functionl component.

The useState hooks allow us to store data in state in a functional component. It can contain strings, numbers, booleans,arrays, objects or combination of any of them.
We can create multiple state hooks to track individual values in functional component.

The useState accepts a initial value and returns two values.
One the current state and other a function that updates the state.

We need to import the useState from react to use it.
Or we can use React.useState. Both of those methods allow us to use useState hook.

So first we will create a functional component file name Counter.js

Here we will create a functional variable UseState. So now name of our variable is different from the file name.

```
const UseState = () => {
}
```

We will export this in our App.js so we will have export default UseState at the end.

```
export default UseState;
```

Now in order to use this function, we will go to App.js and we will import UseState which is name of our function from ./Counter. Here Counter is name of the file from where we can access to our function.

```
import UseState from './Counter';

function App() {
return (
<div>
<UseState />
</div>
);
```

Now we will add our required styling in our App.css . This will contain all styling we need in our Counter.js file.
We already have App.css imported in our App.js so it style all elements from components of counter.js

Now in Counter.js,

we will create a state using useState hook.
Here, we will not import the useState from react but access react to give us its hooks.

```
const [myNum, setMyNum] = React.useState(0);
```

So we have state variable myNum which we have set to have its initial value 0 using useState.
We have function setMyNum which will execute to change the value of our state variable myNum.

Now we want to display the value of state. So we will simply use curly braces to name the state.

```
<p>{myNum}</p>
```

Now on click of a button, we require to change the value of our state. So we will use setMyNum which will be a function that executes to change the state of the function.
Here we want onClick of a button, we want state variable, myNum to increase it value by 1

```
<button class="button" onClick={() => setMyNum(myNum + 1)}>
```

Same way we will create a button that will reduce the value of our state variable by 1. But here, want a condition. We want setMyNum to execute only when value of state variable is greater than 0.

This will restrict counter number to go below 0.

```
<buttonclass="button"
onClick={() => (myNum > 0 ? setMyNum(myNum - 1) : setMyNum(0))}>
```

Hence we create a simple counter using useState hook.

```
const UseState = () => {

const [myNum, setMyNum] = React.useState(0);

return (
<>
<div className="container">
<div className="number">
<p>{myNum}</p>
</div>
<div className="allbuttons">
<button class="button" onClick={() => setMyNum(myNum + 1)}>
INCR
</button>
<button
class="button"
onClick={() => (myNum > 0 ? setMyNum(myNum - 1) : setMyNum(0))}>
DECR
</button>
</div>
</div>
</>
);
};
```