NAMES:

```
import React from "react";
import styled from "styled-components";

const Box = styled.div`
 margin: 10px 10px;
 font-size: 17px;
 background-color: #333333;
 color: white;
 padding: 10px 0;
 padding-right: 45px;
 border-radius: 10px;
 align-items: center;
 text-align: left;
 box-shadow: 10px 10px 5px 0 deeppink;
`;

const List = styled.ul`
 list-style: none;
 text-align: left;
`;

function Names() {
 const names = ["Alice", "Bob", "Charlie", "David"];

 return (
   <Box>
     <List>
       {names.map((name) => (
         <li key={name}>{name}</li>
       ))}
     </List>
   </Box>
 );
}

export default Names;
```

NAMES:

- We define a new React component called Names.
- Inside the Names component, we define an array called names that contains four string elements.

- Inside the List component, we use the map() function to iterate over the names array and create a new <li> element for each name in the array.

- For each <li> element, we set a key prop to the value of the name variable. This is used by React to keep track of each item in the list, and is necessary when rendering lists in React.

NUMBERS:

```
import React from "react";
import styled from "styled-components";

const Box = styled.div`
 margin: 10px 10px;
 font-size: 17px;
 background-color: #333333;
 color: white;
 padding: 10px 0;
 padding-right: 45px;
 border-radius: 10px;
 align-items: center;
 text-align: left;
 box-shadow: 10px 10px 5px 0 deeppink;
`;

const List = styled.ul`
 list-style: none;
 text-align: left;
`;

function Numbers() {
 const numbers = [1, 2, 3, 4, 5];
 const doubledNumbers = numbers.map((number) => number * 2);

 return (
   <Box>
     <List>
       {doubledNumbers.map((number) => (
         <li key={number}>{number}</li>
       ))}
     </List>
   </Box>
 );
}

export default Numbers;
```

NUMBERS:

- Inside the Numbers component, we define an array called numbers that contains five numeric elements.

- We use the map() function to create a new array called doubledNumbers, which contains each number in the numbers array multiplied by 2.

- Inside the List component, we use the map() function to iterate over the doubledNumbers array and create a new <li> element for each number in the array.

- For each <li> element, we set a key prop to the value of the number variable. This is used by React to keep track of each item in the list, and is necessary when rendering lists in React.

ITEMS:

ITEMS:

```
import React, { useState, InputHTMLAttributes } from "react";
import styled from "styled-components";

const Box = styled.div`
 display: flex; /* add display flex */
 flex-direction: column; /* change direction to column */
 justify-content: center; /* center items vertically */
 align-items: center;
 margin: 10px 10px;
 font-size: 17px;
 background-color: #333333;
 color: white;
 padding: 10px 10px;
 padding-right: 45px;
 border-radius: 10px;
 text-align: center;
 box-shadow: 10px 10px 5px 0 deeppink;
`;

const MiniBox = styled.div`
 display: flex;
 flex-direction: row;
 margin: 0 auto;
 width: 100%;
`;

const InputContainer = styled.div`
 display: flex;
 align-items: center;
 justify-content: center;
 margin-right: 10px;
 margin-left: 30px;
`;

const List = styled.ul`
 list-style: none;
 text-align: left;
`;
```

```
type Props = InputHTMLAttributes<HTMLInputElement> & {
 placeholder: string;
 value: string;
 onChange: (event: React.ChangeEvent<HTMLInputElement>) => void;
};

const Input = ({ type, placeholder, value, onChange }: Props) => {
 return (
   <input
     type={type}
     placeholder={placeholder}
     value={value}
     onChange={onChange}
   />
 );
};

function Items() {
 const [items, setItems] = useState(["Rice", "Seaweed",
"Pineapple"]);
 const [inputValue, setInputValue] = useState("");

 const handleAddItem = () => {
   if (inputValue) {
     setItems([...items, inputValue]);
     setInputValue("");
   }
 };

 const handleInputChange = (event:
React.ChangeEvent<HTMLInputElement>) => {
   setInputValue(event.target.value);
 };

 const handleInputKeyDown = (event:
React.KeyboardEvent<HTMLInputElement>) => {
   if (event.key === "Enter") {
     event.preventDefault();
     handleAddItem();
   }
```

- A type alias named "Props" is defined. This is an object type that includes properties inherited from InputHTMLAttributes and three additional properties: placeholder, value, and onChange.

- The InputHTMLAttributes interface is a pre-defined interface in React that defines all the attributes that can be used with the HTML input element. The properties inherited from InputHTMLAttributes include, but are not limited to, autocomplete, autofocus, disabled, name, readonly, required, tabindex, and type. By including InputHTMLAttributes in the type alias definition, the Props type inherits all the properties of InputHTMLAttributes.

- The placeholder property is a string that defines the input field's placeholder text. The value property is a string that defines the current value of the input field. The onChange property is a function that takes an event object and returns void. It is called whenever the input field's value changes.

- A functional component named "Input" is defined. This component takes the "Props" type as its argument and destructures type, placeholder, value, and onChange from it. The component returns an input element with the type, placeholder, value, and onChange properties set to the destructured values.

- A functional component named "Items" is defined. This component is the main component that renders a list of items and an input field to add new items to the list.

- Inside the "Items" component, two state variables are defined using the useState hook. The items variable is an array of strings that stores the items in the list. The inputValue variable is a string that stores the current value of the input field.

- The handleAddItem function is defined, which is called when the user clicks the "Add" button. If the inputValue is not an empty string, a new array is created by spreading the current items array and adding the inputValue to the end of it. The items state is then updated to the new array, and the inputValue state is set to an empty string.

- The handleInputChange function is defined, which is called whenever the input field's value changes. It sets the inputValue state to the current value of the input field.

```
 };

 const handleSubmit = (event: React.FormEvent<HTMLFormElement>) =>
{
   event.preventDefault();
   handleAddItem();
 };

 return (
   <>
     <Box>
       <List>
         {items.map((item, index) => (
           <li key={index}>{item}</li>
         ))}
       </List>{" "}
       <form onSubmit={handleSubmit}>
         <MiniBox>
           <InputContainer>
             <Input
               type="text"
               placeholder="Add an item"
               value={inputValue}
               onChange={handleInputChange}
               onKeyDown={handleInputKeyDown}
             ></Input>
           </InputContainer>

           <button type="submit">Add</button>
         </MiniBox>
       </form>
     </Box>
   </>
 );
}

export default Items;
```

- The handleInputKeyDown function is called whenever a key is pressed while the input field has focus. It checks if the key pressed is the "Enter" key, and if it is, it prevents the default behavior (which is to submit the form) and calls the handleAddItem function to add the current input value to the list.

- The handleSubmit function is called whenever the form is submitted (e.g. by clicking the "Add" button). It prevents the default form submission behavior and calls the handleAddItem function to add the current input value to the list. This is necessary because otherwise clicking the "Add" button would cause the page to reload.

- Inside the "Items" component, the Input component is rendered, passing the type, placeholder, value, and onChange properties to it as props. The handleInputChange function is passed as the onChange prop, so it will be called whenever the input field's value changes.

- The "Add" button is rendered, and the handleAddItem function is called when it is clicked.

APP:

```
import React from "react";
import styled from "styled-components";
import Names from "./components/Names";
import Numbers from "./components/Numbers";
import Items from "./components/Items";

const Container = styled.div`
 display: flex;
 flex-direction: column;
 justify-content: center;
 align-items: center;
`;

const Button = styled.button`
 padding: 5px 20px;
 margin: 10px 10px;
 font-family: monospace;
 font-size: 16px;
 border-radius: 5px;
`;

function App() {
 const names = ["Alice", "Bob", "Charlie", "David"];

 return (
   <Container>
     <Names />
     <Numbers />
     <Items />
   </Container>
 );
}

export default App;
```