```
// Clear the console to start with a clean slate

console.clear();

// Step 1: Create an original object with nested structure

let originalObject = {

    name: "Ajay",

    address: {

        city: "Delhi"  // This is a nested object
    }
};

// Step 2: Create a shallow copy using the spread operator

let shallowCopy = { ...originalObject };

// `shallowCopy` gets its own copy of the top-level keys. But the `address` object is still the SAME reference as in `originalObject`.

// Step 3: Create a deep copy using JSON.parse(JSON.stringify(...))

let deepCopy = JSON.parse(JSON.stringify(originalObject));

// `deepCopy` becomes a completely separate object including a brand-new, independently copied `address` object.

// Step 4: Change the city in the original object's address

originalObject.address.city = "Mumbai";

originalObject.name = "Ashu";

// Because `shallowCopy.address` refers to the same object as in `originalObject`

// This change will also reflect in `shallowCopy`. But `deepCopy.address` remains unchanged — it has its own copy.
```

```javascript

// Step 5: Print the city from the shallow copy

console.log(shallowCopy.address.city);

// Output: "Mumbai"

// Why? Because the `address` in shallowCopy still points to the same object as in `originalObject`, so the update is visible here.

// Step 6: Print the city from the deep copy

console.log(deepCopy.address.city);

// Output: "Delhi"

// Why? Because the deep copy made an independent copy of everything including `address`, so it's unaffected by the change in `originalObject`.
```