PRACTICE ASSESSMENT 6

Task: Create an Express server which will be a simple API with several endpoints.

Build Specifications:

- In `server.js`, construct a simple Express server that will be hosted on port 3000.
    - `server.js` must require the routes module.
- In `routes.js`, create this array of car objects.

```
const cars = [
  { make: "Ford", model: "Mustang Mach-E", price: 46000 },
  { make: "Chevy", model: "Chevelle SS", price: 22000 },
  { make: "Ford", model: "Escape", price: 8000 },
  { make: "Kia", model: "Soul", price: 17500 }
];
```

- In `routes.js`, construct a Router object named routes which has the following endpoints:
    - **GET:** `/cars` - As a response, send back the entire cars array as JSON.
    - **GET:** `/cars/:model` - As a response, send back only one of the car objects as JSON.
        i. It must respond with the car that has the name specified in the route parameter.
        ii. If there is no such car in the array, respond with status code 404 (Not Found.).
    - **POST:** `/cars` - Add a car to the array using the JSON body of the request. Respond with the new car object as JSON and status 201.
    - **GET:** `/cars-limited` - Takes a query string parameter `limit`. As a JSON response, send back an array of the cars, but only include the number of cars indicated by the limit starting with the first car. For example, for `GET /cars-limited?limit=2`, the response will be `[ { make: "Ford", model: "Mustang Mach-E", price: 46000 }, { make: "Chevy", model: "Chevelle SS", price: 22000 } ]` If the limit is greater than the length of the array, just respond with the entire array.
    - **GET:** /cars-search – Takes a query string parameter maxPrice. As a JSON response, send back an array of the cars that a price less than or equal to the maxPrice. If nothing matches, respond with an empty array.