Interactive Challenge / Mini-Project

Your Turn!

1. Create a `TagList` component that:
    - Receives a list of tags and a filter string.
    - Uses `useMemo` to compute the filtered list.
    - Is wrapped in `React.memo` to avoid unnecessary re-renders.
2. Create a `TagInput` component that:
    - Accepts a memoized `onAddTag` callback via `useCallback`.
    - Only re-renders when the callback or input value changes.
3. Show how changing unrelated state in the parent does **not** re-render the memoized `TagList` or `TagInput`.

Your Turn!

1. Create a `ProfileSettings` component that is only loaded when the user clicks a "Settings" button.
2. Use `React.lazy()` and `Suspense` to load the component with a loading spinner.
3. Add a route `/admin` that lazy-loads an `AdminPanel` component only when visited.
4. Show how to handle loading errors with an error boundary.

Your Turn!

1. Use Webpack Bundle Analyzer (or Rsdoctor/Statoscope) on your project.
2. Identify the three largest libraries in your bundle.
3. Refactor your imports to only include what's needed (e.g., for lodash, date-fns, or moment).

4. Change your `tsconfig.json` to `"module": "esnext"` and rerun your build—does the bundle shrink?

5. Remove an unused library and rerun the analyzer—how much did you save?

6. Bonus: Add code splitting for a rarely-used admin page and compare the initial chunk size before and after.

Your Turn!

1. Write a test for a `CommentBox` component that:
    - Renders an input and a “Post” button.
    - Calls a provided `onPost` callback with the input value when clicked.
    - Clears the input after posting.
2. Add a lint rule that forbids `console.log` statements in production code.
3. Debug a failing test: The test expects “Approved!” to appear, but it doesn’t—what could be wrong?

Your Turn!

1. Analyze your current TypeScript project’s build time using `tsc --diagnostics`.
2. Refactor a complex type to a simpler, flatter structure and measure the impact on build time.
3. Enable `"incremental": true` and `"skipLibCheck": true` in your `tsconfig.json` —how much faster are rebuilds?
4. Use Webpack or esbuild to tree shake unused exports—compare the bundle size before and after.
5. Split your project into two packages with project references; measure build and type-checking speed.