Control Structures (`@if`, `@for`, `@each`, `@while`)

Assignment 1: Theme Switcher

Create a variable `$theme` with values `light` and `dark`.
Use `@if` to change:

- Background color
- Text color
- Border color

Assignment 2: Grid System Generator

Using `@for`, generate `.col-1` to `.col-12` classes.
Each class should have:

```
width: (100% / 12) * column-number;
```

Assignment 3: Dynamic Color Utilities

Create a list of colors `(red, green, blue, orange)`
Use `@each` to generate:

```
.text-red
.text-green
.text-blue
.text-orange
```

Assignment 4: Spacing Utility Generator

Using `@while`, generate:

```
.m-1 to .m-5
```

Each class should apply:

```
margin: 5px * number;
```

Inheritance & `@extend`

Assignment 1: Alert System

Create a base class `.alert` with padding and border-radius.
Extend it to create:

- `.alert-success`
- `.alert-danger`
- `.alert-warning`

Assignment 2: Card Variants

Create `%card-base` placeholder.
Extend it for:

- `.card-basic`
- `.card-featured`
- `.card-dark`

Assignment 3: Button Types

Create `.btn` class.
Use `@extend` to create:

- `.btn-primary`
- `.btn-secondary`
- `.btn-outline`

Partials & `@import` / `@use`

Assignment 1: Project Structure

Create partial files:

- `_variables.scss`
- `_mixins.scss`
- `_buttons.scss`

Import them into `style.scss`.

Assignment 2: Theme System

In `_colors.scss` define:

`$primary, $secondary, $danger`

Use `@use` in main file and apply them in components.

Assignment 3: Typography Module

Create `_typography.scss` with:

- Heading styles
- Paragraph styles
  Import into main file.

Built-in Functions

Assignment 1: Color Manipulation

Take a base color.
Create:

- Lighter version (20%)
- Darker version (15%)
- Mixed color with white

Assignment 2: Pricing Layout

Use `percentage()` to create responsive width classes:

```
.w-25
.w-50
.w-75
.w-100
```

Assignment 3: String Utility

Create a variable `$brand-name`.
Use:

- `to-upper-case()`
- `str-length()`

Map Functions

Assignment 1: Button Generator

Create a map:

```
(primary: blue, success: green, danger: red)
```

Generate `.btn-primary`, `.btn-success`, `.btn-danger`.

Assignment 2: Spacing System

Create a spacing map:

```
(small: 5px, medium: 10px, large: 20px)
```

Generate:

```
.p-small
.p-medium
.p-large
```

Assignment 3: Nested Theme Map

Create nested map:

```
light: (bg: white, text: black)
dark: (bg: black, text: white)
```

Apply theme dynamically using `map-get()`.

Assignment 4: Map Validation

Use `map-has-key()` to check if a key exists before applying style.

Selector Functions

Assignment 1: Hover Generator

Create a mixin that takes a selector and adds `:hover` using `selector-append()`.

Assignment 2: Selector Replace

Given `.btn-primary`, replace `primary` with `danger` dynamically.

Assignment 3: Nested Selector Builder

Use `selector-nest()` to combine `.card` and `.title`.

Assignment 4: Selector Unify

Combine `.btn` and `a` using `selector-unify()`.

SCSS Selectors (Nesting, &, Interpolation, Placeholder)

Assignment 1: BEM Structure

Create a `.card` component with:

- `__header`
- `__body`
- `__footer`
- `--featured`

Assignment 2: Theming with Parent Selector

Use:

```
.dark-theme &
```

To change styles dynamically.

Assignment 3: Dynamic Class Generator

Using interpolation:

`$type: "success"`

Generate `.alert-success`.

Assignment 4: Placeholder Usage

Create `%box-style` and extend it in:

- `.product-card`
- `.profile-card`
- `.dashboard-card`

FINAL PRACTICE PROJECT

🎯 Mini SCSS Framework

Build a small utility framework that includes:

Button system (using maps)
Grid system (using @for)
Theme system (nested maps)
Hover generator (selector functions)
Card component (BEM structure)
Spacing utilities (loop)