Equivalence Classes (EC)

1. **EC1: `s1` is a valid prefix of `s2`.**
    - **Example: `s1 = "abc", s2 = "abcdef"`**
    - **Expected Output: `true`**
2. **EC2: `s1` is not a prefix of `s2` but has the same length.**
    - **Example: `s1 = "abc", s2 = "def"`**
    - **Expected Output: `false`**
3. **EC3: `s1` is not a prefix of `s2` and is shorter than `s2`.**
    - **Example: `s1 = "abc", s2 = "xyzdef"`**
    - **Expected Output: `false`**
4. **EC4: `s1` is an empty string.**
    - **Example: `s1 = "", s2 = "abc"`**
    - **Expected Output: `true`**
5. **EC5: `s1` and `s2` are the same string.**
    - **Example: `s1 = "abc", s2 = "abc"`**
    - **Expected Output: `true`**
6. **EC6: `s1` is longer than `s2`.**
    - **Example: `s1 = "abcdef", s2 = "abc"`**
    - **Expected Output: `false`**

Boundary Classes (BC)

Boundary value analysis focuses on testing values at the edges of the input domain.

1. **BC1: `s1` is exactly one character shorter than `s2`.**
    - **Example: `s1 = "abc", s2 = "abcd"`**
    - **Expected Output: `true`**
2. **BC2: `s1` and `s2` are of the same length, and `s1` is a prefix.**
    - **Example: `s1 = "abcd", s2 = "abcd"`**
    - **Expected Output: `true`**
3. **BC3: `s1` and `s2` are of the same length, and `s1` is not a prefix.**
    - **Example: `s1 = "abcd", s2 = "abce"`**
    - **Expected Output: `false`**
4. **BC4: `s1` is exactly one character longer than `s2`.**
    - **Example: `s1 = "abcde", s2 = "abcd"`**

- Expected Output: `false`

5. BC5: First character mismatch between `s1` and `s2`.
   - Example: `s1 = "xbc"`, `s2 = "abc"`
   - Expected Output: `false`
6. BC6: Last character mismatch between `s1` and `s2`.
   - Example: `s1 = "abc"`, `s2 = "abd"`
   - Expected Output: `false`
7. BC7: `s1` is an empty string, and `s2` is non-empty.
   - Example: `s1 = ""`, `s2 = "abc"`
   - Expected Output: `true`
8. BC8: Both `s1` and `s2` are empty strings.
   - Example: `s1 = ""`, `s2 = ""`
   - Expected Output: `true`

Summary of Equivalence Classes and Boundary Classes

Class	Description	Example	Expected Output
EC1	`s1` is a valid prefix of `s2`	`"abc"`, `"abcdef"`	`true`
EC2	`s1` is not a prefix of `s2`, same length	`"abc"`, `"def"`	`false`
EC3	`s1` is shorter than `s2` but not a prefix	`"abc"`, `"xyzdef"`	`false`
EC4	`s1` is an empty string	`""`, `"abc"`	`true`
EC5	`s1` is the same as `s2`	`"abc"`, `"abc"`	`true`
EC6	`s1` is longer than `s2`	`"abcdef"`, `"abc"`	`false`
BC1	`s1` is one character shorter than `s2`	`"abc"`, `"abcd"`	`true`

BC2	**s1 and s2 are the same length, and s1 is a prefix**	`"abcd"`, `"abcd"`	`true`
BC3	**s1 and s2 are the same length, and s1 is not a prefix**	`"abcd"`, `"abce"`	`false`
BC4	**s1 is one character longer than s2**	`"abcde"`, `"abcd"`	`false`
BC5	**First character mismatch between s1 and s2**	`"xbc"`, `"abc"`	`false`
BC6	**Last character mismatch between s1 and s2**	`"abc"`, `"abd"`	`false`
BC7	**s1 is an empty string**	`""`, `"abc"`	`true`
BC8	**Both s1 and s2 are empty strings**	`""`, `""`	`true`

These classes and boundary values ensure the function is rigorously tested across typical, edge, and invalid cases.