Assignment 3: ADS for SUBSTRINGS

You will not be turning in this assignment.
Time allocation (max): thinking, 1 hour; pseudocode, 3 hours

Design an abstract data structure for the data type SUBSTRING.

1. Select a set of *tokens* to represent constants and variables in the domain.

2. Identify the *component parts* (the subtypes) of the data structure, and the functions used to recognize those parts (eg empty-substring, character, substring).

3. Identify the *decomposition axiom* which specifies how to construct and take apart SUBSTRING objects. This axiom will include the definition of *accessor functions* which access parts of a SUBSTRING.

4. Identify a *constructor function* (again defined in the decomposition axiom) which permits building compound substrings out of simple substrings. Consider the difference between a proper substring (A proper subcomponent is a component which is always smaller than its container.)

5. Identify some rules, or *invariants*, which hold between component parts of the data structure. These define the "methods" of the object type.

6. Name some *facts* which are true of this data structure. These define the type hierarchy and the other constraints on the object.

7. Identify the *induction principle* for SUBSTRINGS.

8. Using the induction principle, write pseudocode for some simple recursive functions/methods which implement the invariants of the structure.

9. Finally, suggest some computational data structures which would be appropriate for implementing the SUBSTRING ADS.

Here are some axioms you may need:

The definition of a substring:

```
x sub y  =def=  z1*x*z2 = y
```

The empty string is a substring of every string

```
E sub y
```

No string is a substring of the empty string

```
not(y sub E)
```

Prefixing a character to a string maintains the substring relation

```
if (x sub y) then (x sub u·y)
```

The following three properties of the substring relation establish that *substring is an ordering relation.*

transitivity	if `s1` is a substring of `s2`, and `s2` is a substring of `s3`, then `s1` is a substring of `s3`
antisymmetry	if two strings are substrings of each other, they are equal
reflexivity	a string is a substring of itself

Prove or define the above relations. Then prove:

- A string is a substring of itself when a character is prefixed.
- A string is a substring of the empty string when it is the empty string.
- Substring implies all the characters in the substring are in the string.
- The length of a substring is equal to or less than the length of the string.

Extend the results:

The definition of a **proper** substring:

```
x proper-sub y  =def=  not(z1=E and z2=E) and z1*x*z2 = y
```

Prove the properties of *proper* substrings (transitivity, irreflexivity, asymmetry)