Type Synonyms
What if I want to call int * int * int!
a date?

```haskell
type date = int * int * int!
```

Type Synonyms

**type vs datatype**
Datatype introduces a new type name, distinct from all existing types
```haskell
datatype suit = Club | Diamond | Heart | Spade
datatype rank =
    Jack | Queen | King | Ace | Num of int
```

Type is just another name
```haskell
type card = suit * rank
```

Type Synonym

**Why?**
For now, just for convenience.
It doesn’t let us do anything new.

Later in the course we will see another use related to modularity.
Write a function that appends two string lists...

We expected
\[\text{string list} \times \text{string list} \rightarrow \text{string list}\]

But the type checker says
\[\text{'a list} \times \text{'a list} \rightarrow \text{'a list}\]

Why is this okay?

The type \('a\) is more general

More general types “can be used” as any less general type.

The “more general” rule

A type \(t1\) is more general than the type \(t2\) if you can take \(t1\), replace its type variables consistently, and get \(t2\)
Equality Types

Write a contains function...

Equality Types

Double quotes arise from use of the ‘=’ operator

We can only use ‘=’ on types that can be compared

Generality rules work the same, except substitution must be some type which can be compared with ‘=’

Syntactic Sugar

If-then-else is just a case statement in disguise...

Syntactic Sugar

Pattern matching...