Types
Still on “What types tell us”

- `build-list : natural (natural -> X) -> [X]`
- `filter : (X -> boolean) [X] -> [X]`
- `quicksort : [X] (X X -> boolean) -> [X]`
- `map : (X -> Y) [X] -> [Y]`
- `andmap : (X -> boolean) [X] -> boolean`
- `ormap : (X -> boolean) [X] -> boolean`
- `foldr : (X Y -> Y) Y [X] -> Y`
What does most general type mean?

Consider cross. It can have types:

- `[number] [number] -> [pairOf number number]`
- `[symbol] [symbol] -> [pairOf symbol symbol]`
- `[X] [X] -> [pairOf X X]`
- `[symbol] [number] -> [pairOf symbol number]`
- `[number] [bool] -> [pairOf number bool]`
- `[number] [[bool]] -> [pairOf number [bool]]`
- `[number] [[X]] -> [pairOf number [X]]`
- `[X] [Y] -> [pairOf X Y]`
Variables vs. "any"

- We've talked about types in contracts.
- Following the book, we've talked about this only informally.
- Sometimes we say "any type", sometimes we talk about "a type X".
- The difference appears only when they appear more than once in a type.
Variables vs. "any"

- The difference only affects the contract, not how the function works.
- The contract tells us when a particular function can be used safely when we can't/don't want to look inside the function.
- Contracts have no meaning if we look at the definition of the function ourselves.
- This means types also have no meaning if we look at the definition of a function.
Variables vs. "any"

- The difference: Simple example
- \( (\text{define (id } x) \ x) \)
- Knowing the definition, we know exactly what this function does, But …
- What if we are only told \( \text{id: any->any} \)?
  - What do we know about \( (\text{id 7}) \)?
- What if we are told \( \text{id: X->X} \)?