Trivia

1) a) In a tagged union, we know which of the 2 types we are storing at any given time (via the tag).

In an untagged union, either type could be stored, and we have no way (by default) to tell which one is stored at any given time.

Suppose we have, in C,

```
union IntOrFloat {
    int anInt;
    float aFloat;
}
```

Why is this unsafe?

Because if we use it as a float when it's storing an int, we don't just cast an int to a float; we interpret the bits of the int as a float. This is probably nonsense.

1) b) $A \uplus B$, the disjoint union.

c) $A \cup B$

2) Because Agda is pure.

3) You can write for more specific types, for instance, a type of lists which have length as part of their type, so we can reason about lists of particular length.

$$\frac{(E_2, \Sigma, \sigma) \longrightarrow (E_2', \Sigma, \sigma)}{(E_1 \oslash E_2, \Sigma, \sigma) \longrightarrow (E_1 \oslash E_2', \Sigma, \sigma)}$$ division-operator-right

$$\frac{}{(E_1 \oslash 0, \Sigma, \sigma) \longrightarrow (\text{error}, \Sigma, \sigma)}$$ division-operator-error

$$\frac{(E_1, \Sigma, \sigma) \longrightarrow (E_1', \Sigma, \sigma) \qquad n \neq 0}{(E_1 \oslash n, \Sigma, \sigma) \longrightarrow (E_1' \oslash n, \Sigma, \sigma)}$$ division-operator-left

$$\frac{n_2 \neq 0 \qquad n = \overbrace{n_1 \oslash n_2}^{\text{mathematical statement, i.e. a number}}}{(\underbrace{n_1 \oslash n_2}_{\text{Expression}}, \Sigma, \sigma) \longrightarrow (n, \Sigma, \sigma)}$$ division-operator-apply

$$\frac{(E_1, \Sigma, \sigma) \Downarrow n \quad (E_2, \Sigma, \sigma) \Downarrow m \quad p = n \oslash m \quad m \neq 0}{(E_1 \oslash E_2, \Sigma, \sigma) \Downarrow p}\ \text{division-op}$$

$$\frac{(E_2, \Sigma, \sigma) \Downarrow 0}{(E_1 \oslash E_2, \Sigma, \sigma) \Downarrow \text{error}}\ \text{division-op-error}$$