Isabelle/HOL Exercises
Lists

Recursive Functions and Induction: Zip

Read the chapter about total recursive functions in the "Tutorial on Isabelle/HOL" (*fun*, Chapter 3.5).

In this exercise you will define a function *Zip* that merges two lists by interleaving. Examples: *Zip [a1, a2, a3] [b1, b2, b3] = [a1, b1, a2, b2, a3, b3]* and *Zip [a1] [b1, b2, b3] = [a1, b1, b2, b3]*.

Use three different approaches to define *Zip*:

1. by primitive recursion on the first list,
2. by primitive recursion on the second list,
3. by total recursion (using *fun*).

```
consts zip1 :: "'a list ⇒ 'a list ⇒ 'a list"
consts zip2 :: "'a list ⇒ 'a list ⇒ 'a list"
consts zipr :: "'a list ⇒ 'a list ⇒ 'a list"
```

Show that all three versions of *Zip* are equivalent.

Show that *zipr* distributes over *append*.

```
lemma "⟦length p = length u; length q = length v⟧ ⟹
  zipr (p@q) (u@v) = zipr p u @ zipr q v"
```

Note: For *fun*, the order of your equations is relevant. If equations overlap, they will be disambiguated before they are added to the logic. You can have a look at these equations using *thm zipr.simps*.