Isabelle/HOL Exercises
Lists

Recursive Functions and Induction: Zip

Read the chapter about recursive definitions in the "Tutorial on Isabelle/HOL" (*recdef*, 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 *recdef*).

```
consts zip1 :: "'a list ⇒ 'a list ⇒ 'a list"
consts zip2 :: "'a list ⇒ 'a list ⇒ 'a list"
consts zipr :: "('a list × 'a list) ⇒ 'a list"

primrec
  "zip1 []     ys = ys"
  "zip1 (x#xs) ys = (case ys of [] ⇒ (x#xs) | z#zs ⇒ x#z#(zip1 xs zs))"

primrec
  "zip2 xs []     = xs"
  "zip2 xs (y#ys) = (case xs of [] => (y#ys) | z#zs => z # y # zip2 zs ys)"

recdef zipr "measure (λ(xs,ys). length xs + length ys)"
  "zipr ([],ys)     = ys"
  "zipr (xs,[])     = xs"
  "zipr ((x#xs),ys) = x#zipr(ys,xs)"
```

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

```
lemma "∀ ys. zip1 xs ys = zip2 xs ys"
  apply (induct xs)
```

```
    apply auto
    apply (case_tac ys)
    apply auto
  apply (case_tac ys)
  apply auto
done

lemma [simp]: "zipr (xs,[]) = xs"
  apply (case_tac xs)
  apply auto
done

lemma [simp]: "zipr ((x#xs),ys) = x#zipr(ys,xs)"
  apply (case_tac ys)
  apply auto
done

lemma "∀ xs. zip2 xs ys = zipr (xs,ys)"
  apply (induct ys)
    apply auto
  apply (case_tac xs)
    apply auto
done

lemma "∀ ys. zipr (xs,ys) = zip1 xs ys"
  apply (induct xs)
    apply auto
  apply (case_tac ys)
    apply auto
done
```

Show that *zipr* distributes over *append*.

```
lemma zipr_append:
  "∀u q v. length p = length u ∧  length q = length v ⟶
  zipr(p@q,u@v) = zipr(p,u) @ zipr(q,v)"
  apply (induct p)
    apply auto
  apply (case_tac u)
    apply auto
done

lemma "⟦length p = length u; length q = length v⟧ ⟹
  zipr(p@q,u@v) = zipr(p,u) @ zipr(q,v)"
  by (simp add: zipr_append)
```

Note: For *recdef*, 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*.