Isabelle/HOL Exercises
Lists

Summation, Flattening

Define a function *sum*, which computes the sum of elements of a list of natural numbers.

```
sum :: "nat list ⇒ nat"
```

Then, define a function *flatten* which flattens a list of lists by appending the member lists.

```
flatten :: "'a list list ⇒ 'a list"
```

Test your functions by applying them to the following example lists:

```
lemma "sum [2::nat, 4, 8] = x"
lemma "flatten [[2::nat, 3], [4, 5], [7, 9]] = x"
```

Prove the following statements, or give a counterexample:

```
lemma "length (flatten xs) = sum (map length xs)"
lemma sum_append: "sum (xs @ ys) = sum xs + sum ys"
lemma flatten_append: "flatten (xs @ ys) = flatten xs @ flatten ys"
lemma "flatten (map rev (rev xs)) = rev (flatten xs)"
lemma "flatten (rev (map rev xs)) = rev (flatten xs)"
lemma "list_all (list_all P) xs = list_all P (flatten xs)"
lemma "flatten (rev xs) = flatten xs"
lemma "sum (rev xs) = sum xs"
```

Find a (non-trivial) predicate *P* which satisfies

```
lemma "list_all P xs ⟶ length xs ≤ sum xs"
```

Define, by means of primitive recursion, a function *list_exists* which checks whether an element satisfying a given property is contained in the list:

```
list_exists :: "('a ⇒ bool) ⇒ ('a list ⇒ bool)"
```

Test your function on the following examples:

```
lemma "list_exists (λ n. n < 3) [4::nat, 3, 7] = b"
lemma "list_exists (λ n. n < 4) [4::nat, 3, 7] = b"
```

Prove the following statements:

```
lemma list_exists_append:
```

```
  "list_exists P (xs @ ys) = (list_exists P xs ∨ list_exists P ys)"
```
lemma *"list_exists (list_exists P) xs = list_exists P (flatten xs)"*

You could have defined *list_exists* only with the aid of *list_all*. Do this now, i.e. define a function *list_exists2* and show that it is equivalent to *list_exists*.