Exercise 1 (Recursion on numbers and lists)**.**

1. Define a function which computes the factorial $n!$ of a given natural number n.
2. Define a function `prime (n : nat) : bool` which checks if a number is prime.

   Hint. Use a local helper function.
3. What is wrong with the following definition?

```
Fixpoint half (n : nat) : nat :=
  if Nat.ltb n 2 then 0 else half (n - 2) + 1.
```

   How can you reformulate the definition so that it is accepted by Coq?
4. Define a polymorphic function which computes the last element of a list. What is the result of your function on an empty list?
5. A *suffix* of a list l is any list which can be obtained from l by removing some $n \geq 0$ initial elements. For example, the suffixes of $[1;2;3]$ are: $[1;2;3]$, $[2;3]$, $[3]$ and $[]$.

   Define a function which given a list l computes the list of all suffixes of l in the order of decreasing length.

Exercise 2 (Higher-order functions)**.**

1. Recall the type of binary trees from the lecture.

```
Inductive tree A := leaf (x : A) | node (l r : tree A).
```

   Define appropriate `map` and `fold` functions for such trees. The `map` function should apply a given function to all elements in the leaves. The `fold` function should accumulate the elements in the leaves with a function given as an argument.
2. Using your `fold` function from the previous point, define a function which converts a tree into a list by accumulating all elements in the leaves from left to right.