Before beginning to solve the exercises, open a new theory file named Ex01.thy and add the following three lines at the beginning of this file.

```
theory Ex01
imports Main
begin
```

Exercise 1.1 Calculating with natural numbers

Use the `value` command to turn Isabelle into a fancy calculator and evaluate the following natural number expressions:

\[ 2 + (2::nat) \]
\[ (2::nat) \times (5 + 3) \]
\[ (3::nat) \times 4 - 2 \times (7 + 1) \]

Can you explain the last result?

Exercise 1.2 Natural number laws

Formulate and prove the well-known laws of commutativity and associativity for addition of natural numbers.

Exercise 1.3 Counting elements of a list

Define a function which counts the number of occurrences of a particular element in a list.

\[ \text{fun count :: } "a \text{ list } \Rightarrow 'a \Rightarrow \text{ nat}" \]

Test your definition of `count` on some examples and prove that the results are indeed correct.

Prove the following inequality (and additional lemmas if necessary) about the relation between `count` and `length`, the function returning the length of a list.

\[ \text{theorem } "\text{count } xs \ x \leq \text{length } xs" \]
Exercise 1.4  Adding elements to the end of a list

Recall the definition of lists from the lecture. Define a function \texttt{snoc} that appends an element at the right end of a list. Do not use the existing append operator \texttt{@} for lists.

\textbf{fun} \texttt{snoc} :: “\texttt{′a list ⇒ ′a ⇒ ′a list}”

Convince yourself on some test cases that your definition of \texttt{snoc} behaves as expected, for example run:

\textbf{value} “\texttt{snoc [] c}”

Also prove that your test cases are indeed correct, for instance show:

\textbf{lemma} “\texttt{snoc [] c = [c]}”

Next define a function \texttt{reverse} that reverses the order of elements in a list. (Do not use the existing function \texttt{rev} from the library.) Hint: Define the reverse of \texttt{x # xs} using the \texttt{snoc} function.

\textbf{fun} \texttt{reverse} :: “\texttt{′a list ⇒ ′a list}”

Demonstrate that your definition is correct by running some test cases, and proving that those test cases are correct. For example:

\textbf{value} “\texttt{reverse [a, b, c]}”

\textbf{lemma} “\texttt{reverse [a, b, c] = [c, b, a]}”

Prove the following theorem. Hint: You need to find an additional lemma relating \texttt{reverse} and \texttt{snoc} to prove it.

\textbf{theorem} “\texttt{reverse (reverse xs) = xs}”

Homework 1.1  More Finger Exercise with Lists

\textit{Submission until Tuesday, October 28, 10:00am.}

\textbf{Submission Instructions}

Submissions are handled via https://competition.isabelle.systems/.

- Register an account in the system and send the tutor an e-mail with your username.
- Select the competition “Semantics 2019/20” and submit your solution following the instructions on the website.
- The system will check that your solution can be loaded in Isabelle2019 without any errors and reports how many of the main theorems you were able to prove.
- You can upload multiple times; the last upload before the deadline is the one that will be graded.
• If you have any problems uploading, or if the submission seems to be rejected for reasons you cannot understand, please contact the tutor.

General hints:

• If you cannot prove a lemma, that you need for a subsequent proof, assume this lemma by using sorry.

• Define the functions as simply as possible. In particular, do not try to make them tail recursive by introducing extra accumulator parameters — this will complicate the proofs!

• All proofs should be straightforward, and take only a few lines.

Define a function \textit{list\_sum} that computes the sum of the elements in a list of natural numbers:

\texttt{fun list\_sum :: “nat list ⇒ nat”}

Prove that the sum of a list is invariant under reversing the list:

\texttt{theorem list\_sum\_reverse:}
\texttt{“list\_sum (reverse xs) = list\_sum xs”}

\textit{Hint:} You may need a lemma about \textit{snoc} and \textit{list\_sum}.

Define a function \textit{upto} such that \textit{upto} \textit{n} returns the list consisting of the elements \([0..n]\):

\texttt{fun upto :: “nat ⇒ nat list”}

Finally, prove Gauss’ well-known theorem about the sum of the first \textit{n} natural numbers:

\texttt{theorem gauss:}
\texttt{“list\_sum (upto n) = n * (n + 1) div 2”}