Goals  You get a better overview over correctness proofs and actively use abstractions.

Before  Use the same setup of SimplC as in the previous exercise.

Exercise 1 [4] Copy of an Array
The following code obviously copies all elements of an array \( a \) into an array \( b \).

```ml
i = 0;
while (i < n) {
    b[i] = a[i];
    i = i + 1;
}
```

Prove that this is true.

1. Write down the appropriate pre- and postconditions. Use the \texttt{elems} abstractions from the lecture.
2. Specify the loop invariant and verify the code.

Exercise 2 [6] Reverse an Array
The following code reverses the elements of an array (analogue to \texttt{partition} from the lecture).

```ml
i = 0;
j = n;
while (i < j) {
    t = a[i];
    a[i] = a[j-1];
    a[j-1] = t;
    i = i + 1;
    if (i < j)
        j = j - 1;
}
```

Prove that the precondition

\[
\begin{align*}
n &\leq \text{length } a \land \text{elems } a \ 0 \ n = A \land 0 \leq n
\end{align*}
\]

implies the following postcondition:

\[
\begin{align*}
\text{elems } a \ 0 \ n = \text{rev } A
\end{align*}
\]

For the main part of the invariant, you can use

\[
A = \text{rev } (\text{elems } a \ j \ n) @ \text{elems } a \ i \ j @ \text{rev } (\text{elems } a \ 0 \ i)
\]

Hint: For this exercise, \texttt{simp add: elems_split_first} does too much. Use \texttt{subst} to apply the lemmas \texttt{elems_split_first} and \texttt{elems_split_last} in a controlled way. Solve the conditions with \texttt{assumption} and \texttt{simp}. 