Presentation of Mini-Projects
You are invited, on a voluntary basis, to give a short presentation of your mini-projects in the tutorial on July 28.
Depending on how many presentations we have, the time slots will be 5 to 10 minutes, plus 2 minutes for questions.
If you are interested, please write me a short email until Thursday, July 27.

Exercise 13.1 Double-Ended Queues
Design a double-ended queue where all operations have constant-time amortized complexity. Prove functional correctness and constant-time amortized complexity.
For your proofs, it is enough to count the number of newly allocated list cells. You may assume that operations $\text{rev } xs$ and $xs @ ys$ allocate $O(|xs|)$ cells.
Explanation: A double-ended queue is like a queue with two further operations: Function $\text{enq_front}$ adds an element at the front (whereas $\text{enq}$ adds an element at the back). Function $\text{deq_back}$ removes an element at the back (whereas $\text{deq}$ removes an element at the front). Here is a formal specification where the double ended queue is just a list:

abbreviation (input) \text{enq_list} :: "'$a \Rightarrow 'a \ list \Rightarrow 'a \ list$" where
\text{enq_list } x \ xs \equiv \ xs @ [x]"

abbreviation (input) \text{enq_front_list} :: "'$a \Rightarrow 'a \ list \Rightarrow 'a \ list$" where
\text{enq_front_list } x \ xs \equiv x \# \ xs"

abbreviation (input) \text{deq_list} :: "'$a \ list \Rightarrow 'a \ list$" where
\text{deq_list } xs \equiv \text{tl } xs"

abbreviation (input) \text{deq_back_list} :: "'$a \ list \Rightarrow 'a \ list$" where
\text{deq_back_list } xs \equiv \text{butlast } xs"

Hint: You may want to start with the Queue implementation in Thys/Amortized_Examples.

lemma "list_of init = []"
lemma "list_of(enq \ x \ q) = enq_list \ x \ (list_of \ q)"
lemma "list_of(enq_front \ x \ q) = enq_front_list \ x \ (list_of \ q)"
lemma \textquotedblleft list\_of \( q \neq \[] \implies \text{list\_of}(\text{deq} \ q) = \text{deq\_list}\ (\text{list\_of} \ q)\textquotedblright

lemma \textquotedblleft list\_of \( q \neq \[] \implies \text{list\_of}(\text{deq\_back} \ q) = \text{deq\_back\_list}\ (\text{list\_of} \ q)\textquotedblright

**Homework 13  **Pairing Heap

*Submission until Friday, 28. 07. 2017, 11:59am.*

The datatype of pairing heaps defined in the theory Thys/Pairing_Heap comes with the unstated invariant that *Empty* occurs only at the root. We can avoid this invariant by a slightly different representation:

**datatype** \( \text{'}a \text{ hp} = \text{Hp} \text{'}a \) \( \text{linorder list}\)\n
**type synonym** \( \text{'}a \text{ heap} = \text{'}a \text{ hp option}\)

Carry out the development with this new representation. Restrict yourself to the \textit{get\_min} and \textit{delete\_min} operations. That is, define the following functions (and any auxiliary function required)

\textbf{fun} get\_min :: \( \text{'(}a \text{:: linorder) heap} \rightarrow \text{'}a\) \textbf{where}  
\textbf{fun} del\_min :: \( \text{'(}a \text{:: linorder heap} \rightarrow \text{'}a \text{ heap}\) \textbf{where}  
\textbf{fun} php :: \( \text{'(}a \text{:: linorder) hp} \rightarrow \text{bool}\) \textbf{where}  
\textbf{fun} mset\_hp :: \( \text{'(}a \text{ hp} \rightarrow \text{'}a \text{ multiset}\) \textbf{where}  
\textbf{fun} mset\_heap :: \( \text{'}a \text{ heap} \rightarrow \text{'}a \text{ multiset}\) \textbf{where}

and prove the following functional correctness theorems and any lemmas required, but excluding preservation of the invariant:

\textbf{theorem} get\_min\_in: \textit{get\_min} (Some \( h \)) \( \in \text{set\_hp}(\text{hp})\)
\textbf{lemma} get\_min\_min: \( \{ \text{\_php} \ h; \ x \in \text{set\_hp}(\text{hp}) \} \implies \text{get\_min} \ (\text{Some} \ h) \leq \text{x}\)
\textbf{lemma} mset\_del\_min: \textit{mset\_heap} (\text{del\_min} (\text{Some} \ h)) = \text{mset\_hp} \ h - \{\#\text{get\_min}(\text{Some} \ h)\}\)

It is recommended to start with the original theory and modify it as much as needed. 

Note that function \textit{set\_hp} is defined automatically by the definition of type \textit{hp}.