Goals  Manipulate lists on the heap and use ghost variables.


The cons operation (insert a new element at the beginning of a list) is easy to implement on the heap. One just needs to set the next-pointer of the new node to the begin of the existing list. Moreover, the old list stays unchanged, which makes this operation so useful for functional languages.

\[
\text{pre: } \text{list node-alloc node-next p XS NULL } \land \text{ node-alloc q}
\]
\[
\text{post: } \text{list node-alloc node-next q (q # XS) NULL } \land \\
\text{list node-alloc node-next p XS NULL}
\]

\{ * 
\q\rightarrow\text{next} = p; 
* \}

Unfortunately, the given pre-condition is too weak. Insert the necessary additional assumptions and prove the correctness. (Have a look at the “Cheat Sheet” from the lecture).

Exercise 2 [4] Test for an Element

The following program tests whether the value \(v\) is contained in the list \(p\). Prove the program correct. For this, annotate the program with ghost variables, so you do not have to provide witnesses for existential quantifiers manually.

\[
\text{pre: } \text{list node-alloc node-next p XS NULL } \land \text{ DS = list-data node-data XS}
\]
\[
\text{post: } r \neq 0 \iff v \in \text{set DS}
\]

\{ * 
\q = 0; 
/*@ True */
\text{while (p \neq \text{null } \&\& r == 0) } 
\{ 
\q\rightarrow\text{data} == v \{ 
\q = 1; 
\} 
\p = \p\rightarrow\text{next}; 
\} 
* \}

Exercise 3 [5] Insert an Element (Hard)

Verify the following code which inserts a node \(q\) at the position \(pos\) of the list \(p\) and returns the modified list in \(r\).

\[
\text{if (pos == 0) } 
\{ 
\q\rightarrow\text{next} = p; 
\q = q; 
\} \text{ else } 
\{ 
\q = p; 
\text{prev} = p; 
\p = \p\rightarrow\text{next}; 
\pos = \pos - 1; 
\text{while (0 < pos) } 
\{ 
\text{prev} = p; 
\p = \p\rightarrow\text{next}; 
\} 
\} 
\]
pos = pos - 1;
}
prev->next = q;
q->next = p;
}