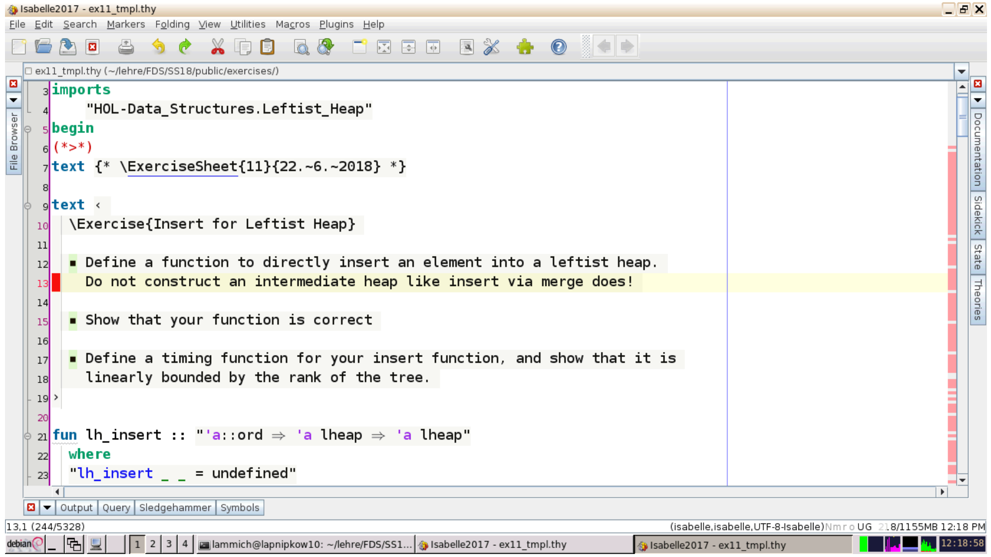
# Script generated by TTT

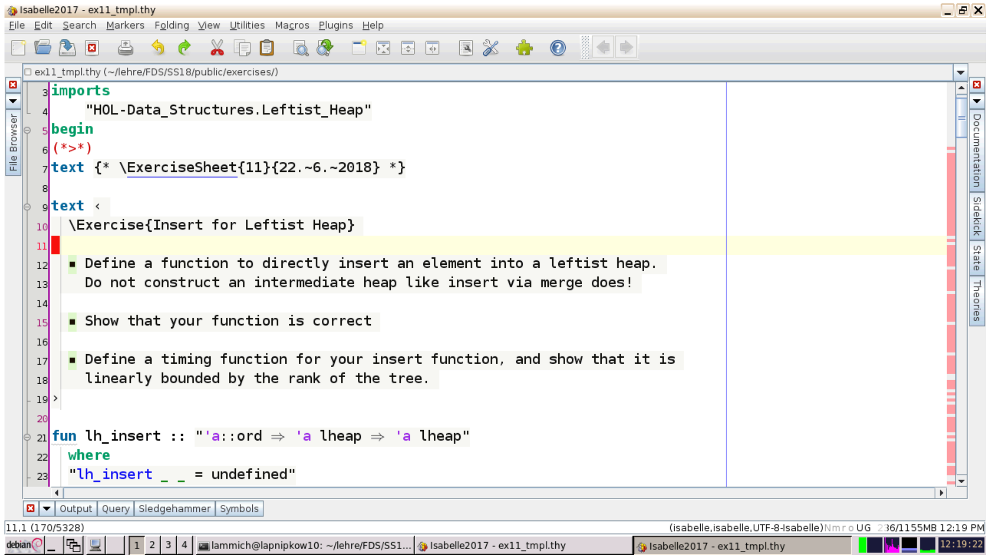
Title: Lammich: FDS Tutorial (22.06.2018)

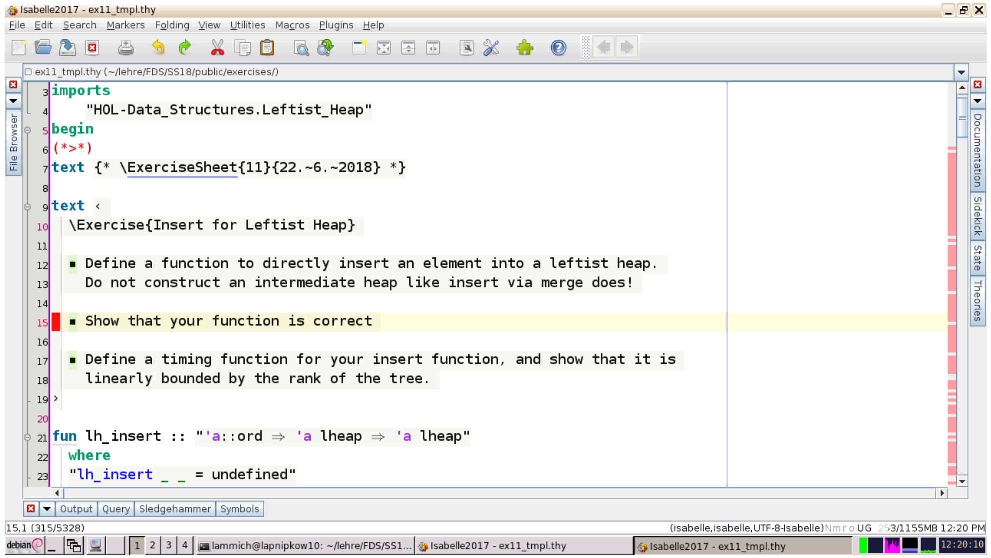
Date: Fri Jun 22 12:18:58 CEST 2018

Duration: 92:56 min

Pages: 106



Isabelle2017 - ex11_tmpl.thy

```
  ▪ Show that your function is correct

  ▪ Define a timing function for your insert function, and show that it is
    linearly bounded by the rank of the tree.
›

fun lh_insert :: "'a::ord ⇒ 'a lheap ⇒ 'a lheap"
  where
  "lh_insert _ _ = undefined"

lemma mset_lh_insert: "mset_tree (lh_insert x t) = mset_tree t + {# x #}"
  sorry

lemma "heap t ⟹ heap (lh_insert x t)"
  sorry

lemma "ltree t ⟹ ltree (lh_insert x t)"
  sorry


fun t_lh_insert :: "'a::ord ⇒ 'a lheap ⇒ nat"
```

Isabelle2017 - Leftist_Heap.thy

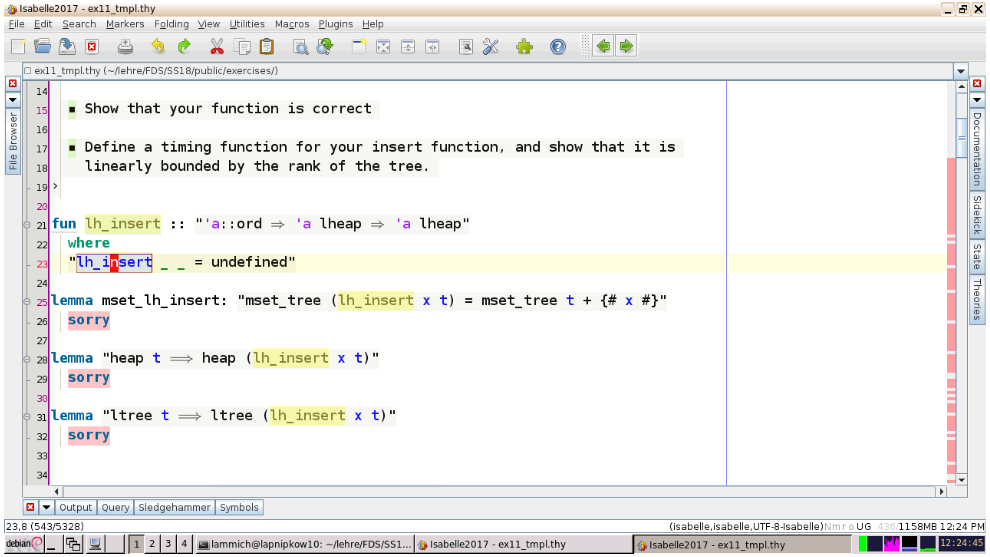
```
fun rk :: "'a lheap ⇒ nat" where
"rk Leaf = 0" |
"rk (Node n _ _ _) = n"

text{* The invariants: *}

fun (in linorder) heap :: "('a,'b) tree ⇒ bool" where
"heap Leaf = True" |
"heap (Node _ l m r) =
  (heap l ∧ heap r ∧ (∀x ∈ set_mset(mset_tree l + mset_tree r). m ≤ x))"

fun ltree :: "'a lheap ⇒ bool" where
"ltree Leaf = True" |
"ltree (Node n l a r) =
 (n = rank r + 1 ∧ rank l ≥ rank r ∧ ltree l & ltree r)"

definition node :: "'a lheap ⇒ 'a ⇒ 'a lheap ⇒ 'a lheap" where
"node l a r =
 (let rl = rk l; rr = rk r
  in if rl ≥ rr then Node (rr+1) l a r else Node (rl+1) r a l)"
```

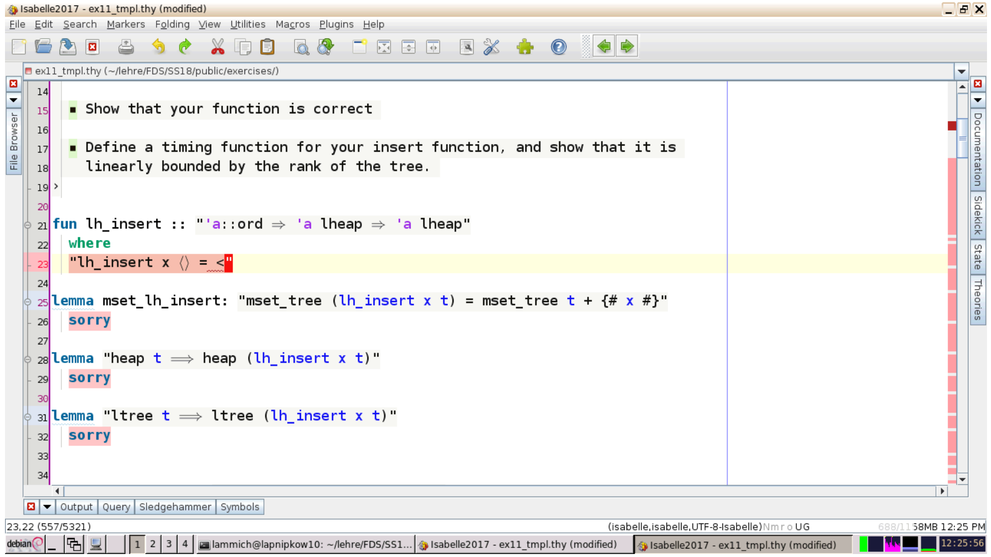
Isabelle2017 - Leftist_Heap.thy

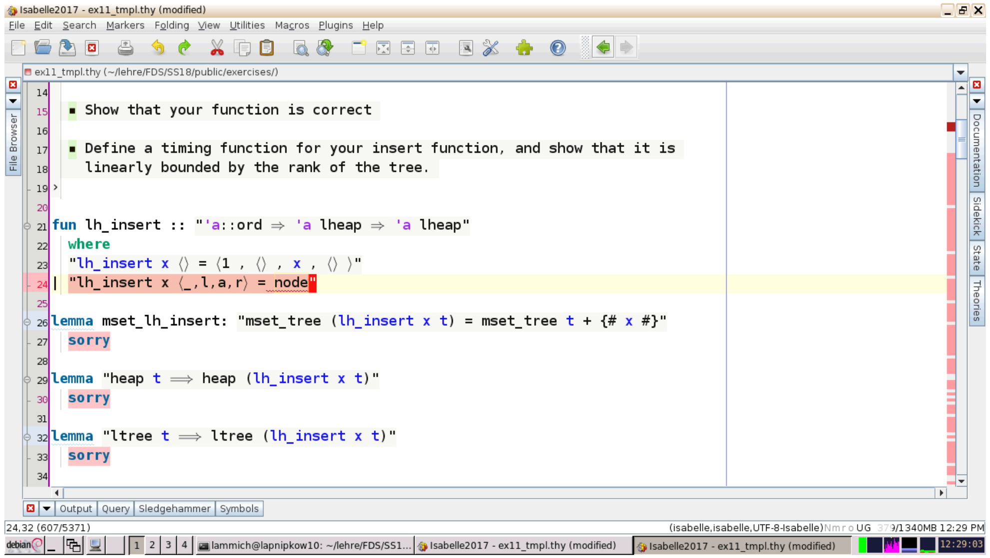
```
text{* The invariants: *}

fun (in linorder) heap :: "('a,'b) tree ⇒ bool" where
"heap Leaf = True" |
"heap (Node _ l m r) =
  (heap l ∧ heap r ∧ (∀x ∈ set_mset(mset_tree l + mset_tree r). m ≤ x))"

fun ltree :: "'a lheap ⇒ bool" where
"ltree Leaf = True" |
"ltree (Node n l a r) =
 (n = rank r + 1 ∧ rank l ≥ rank r ∧ ltree l & ltree r)"

definition node :: "'a lheap ⇒ 'a ⇒ 'a lheap ⇒ 'a lheap" where
"node l a r =
 (let rl = rk l; rr = rk r
  in if rl ≥ rr then Node (rr+1) l a r else Node (rl+1) r a l)"

fun get_min :: "'a lheap ⇒ 'a" where
"get_min(Node n l a r) = a"

text ‹For function ‹merge›:›
```

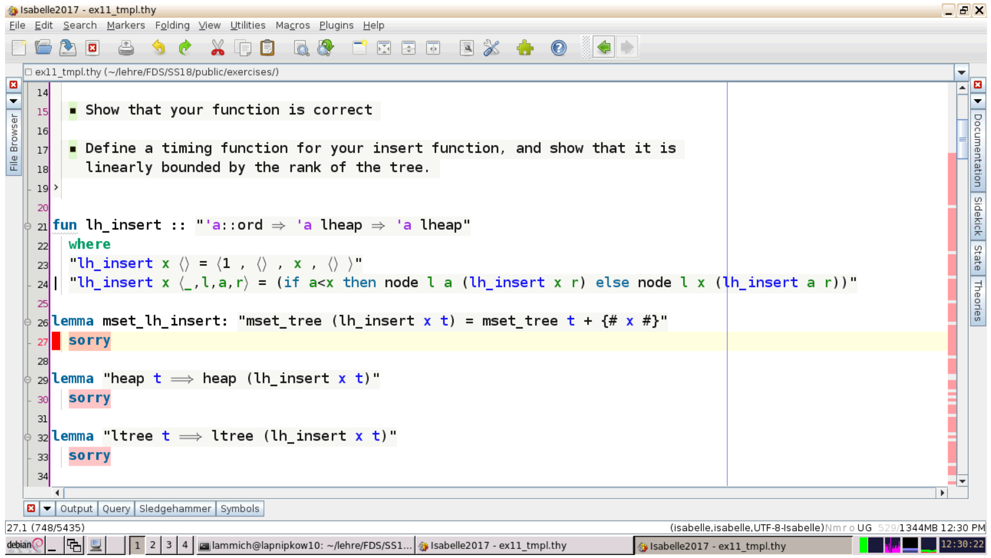
Isabelle2017 - Leftist_Heap.thy

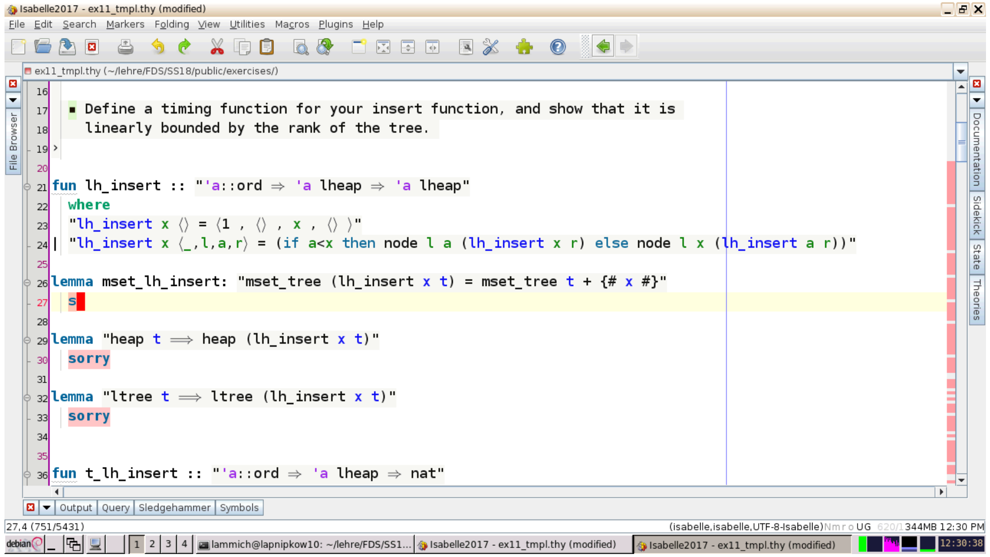
```
subsection "Lemmas"

lemma mset_tree_empty: "mset_tree t = {#} ⟷ t = Leaf"
by(cases t) auto

lemma rk_eq_rank[simp]: "ltree t ⟹ rk t = rank t"
by(cases t) auto

lemma ltree_node: "ltree (node l a r) ⟷ ltree l ∧ ltree r"
by(auto simp add: node_def)

lemma heap_node: "heap (node l a r) ⟷
  heap l ∧ heap r ∧ (∀x ∈ set_mset(mset_tree l + mset_tree r). a ≤ x)"
by(auto simp add: node_def)


subsection "Functional Correctness"

lemma mset_merge: "mset_tree (merge h1 h2) = mset_tree h1 + mset_tree h2"
by (induction h1 h2 rule: merge.induct) (auto simp add: node_def ac_simps)
```

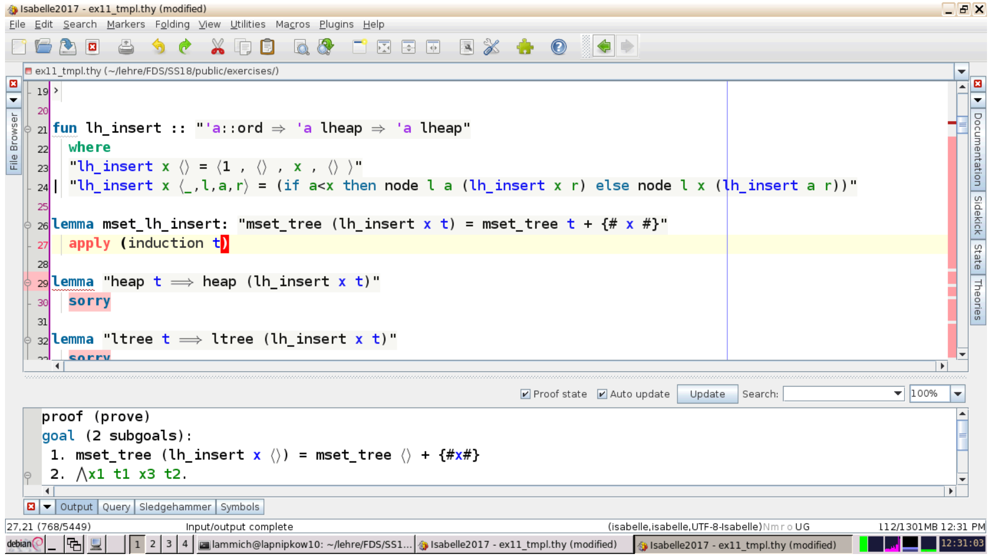

Isabelle2017 - ex11_tmpl.thy
File Edit Search Markers Folding View Utilities Macros Plugins Help
ex11_tmpl.thy (~/lehre/FDS/SS18/public/exercises/)
▪ Show that your function is correct
▪ Define a timing function for your insert function, and show that it is
linearly bounded by the rank of the tree.
›
fun lh_insert :: "'a::ord ⇒ 'a lheap ⇒ 'a lheap"
where
"lh_insert _ _ = undefined"
lemma mset_lh_insert: "mset_tree (lh_insert x t) = mset_tree t + {# x #}"
sorry
lemma "heap t ⟹ heap (lh_insert x t)"
sorry
lemma "ltree t ⟹ ltree (lh_insert x t)"
sorry
Output Query Sledgehammer Symbols
23.8 (543/5328)
(isabelle,isabelle,UTF-8-Isabelle) UG 1158MB 12:24 PM

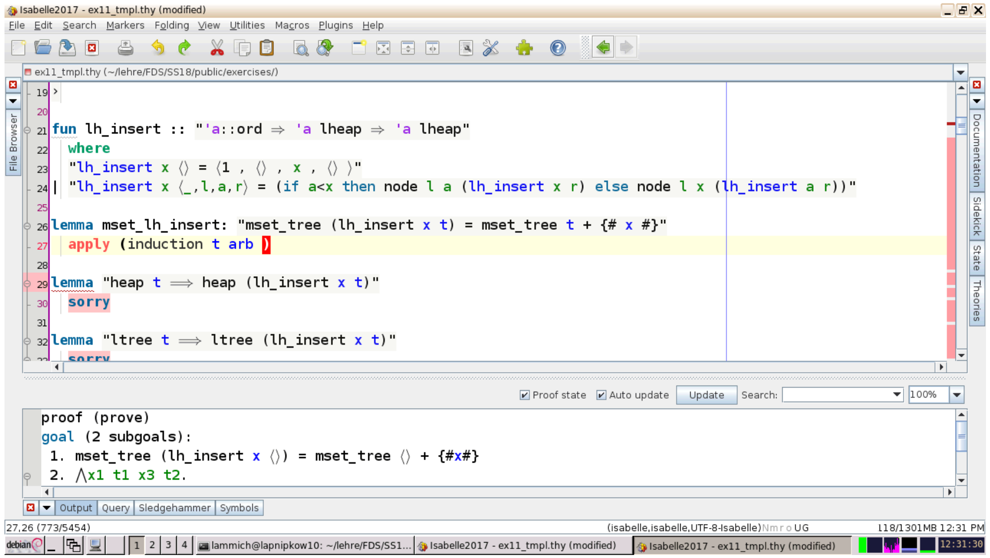

Isabelle2017 - ex11_tmpl.thy (modified)
File Edit Search Markers Folding View Utilities Macros Plugins Help
ex11_tmpl.thy (~/lehre/FDS/SS18/public/exercises/)
▪ Show that your function is correct
▪ Define a timing function for your insert function, and show that it is
linearly bounded by the rank of the tree.
›
fun lh_insert :: "'a::ord ⇒ 'a lheap ⇒ 'a lheap"
where
"lh_insert x ⟨⟩ = ⟨"
lemma mset_lh_insert: "mset_tree (lh_insert x t) = mset_tree t + {# x #}"
sorry
lemma "heap t ⟹ heap (lh_insert x t)"
sorry
lemma "ltree t ⟹ ltree (lh_insert x t)"
sorry
Output Query Sledgehammer Symbols
23.22 (557/5321)
(isabelle,isabelle,UTF-8-Isabelle) UG 12:25 PM

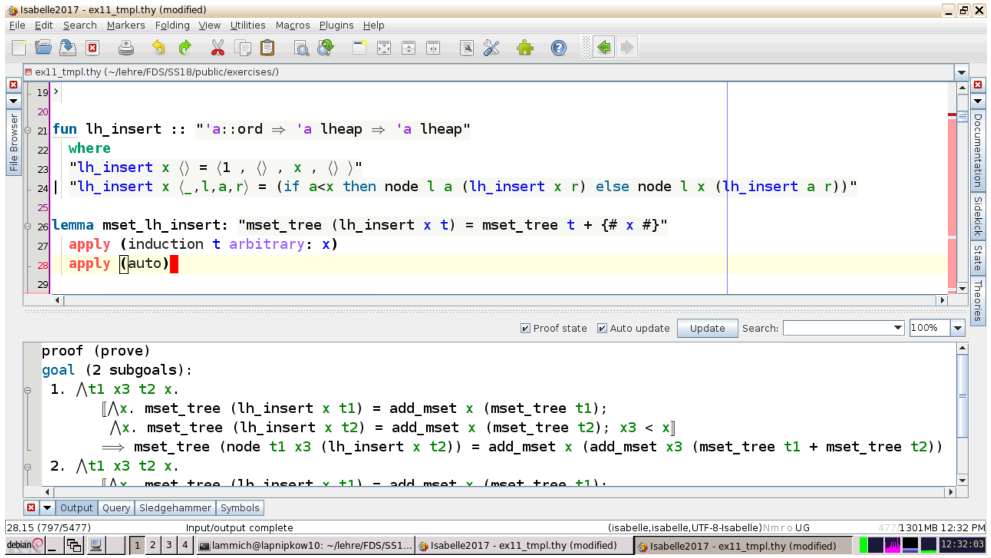

Isabelle2017 - ex11_tmpl.thy (modified)
File Edit Search Markers Folding View Utilities Macros Plugins Help
ex11_tmpl.thy (~/lehre/FDS/SS18/public/exercises/)
▪ Show that your function is correct
▪ Define a timing function for your insert function, and show that it is
linearly bounded by the rank of the tree.
›
fun lh_insert :: "'a::ord ⇒ 'a lheap ⇒ 'a lheap"
where
"lh_insert x ⟨⟩ = ⟨1 , ⟨⟩ , x , ⟨⟩ ⟩"
"lh_insert x ⟨_,l,a,r⟩ = node"
lemma mset_lh_insert: "mset_tree (lh_insert x t) = mset_tree t + {# x #}"
sorry
lemma "heap t ⟹ heap (lh_insert x t)"
sorry
lemma "ltree t ⟹ ltree (lh_insert x t)"
sorry
Output Query Sledgehammer Symbols
24.32 (607/5371)
(isabelle,isabelle,UTF-8-Isabelle) UG 1340MB 12:29 PM

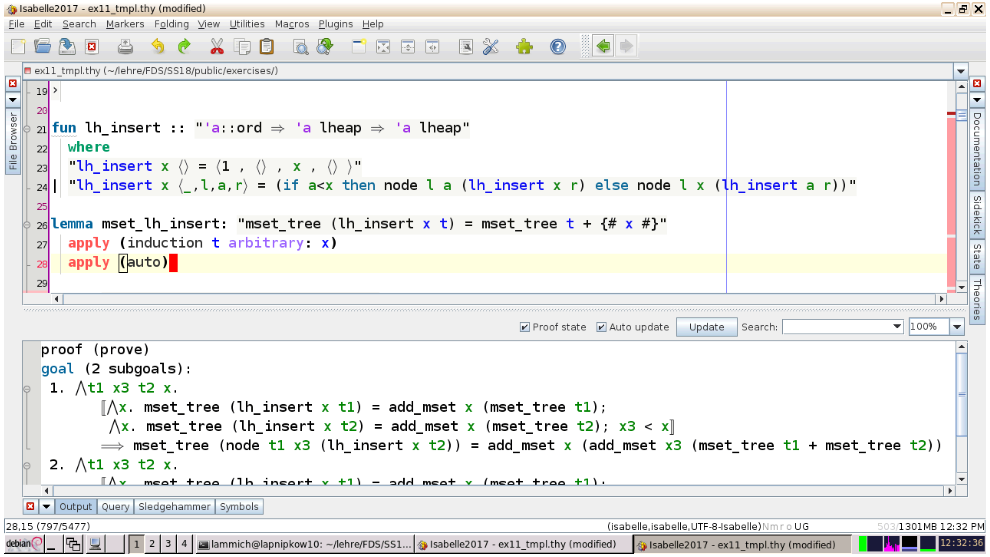

Isabelle2017 - ex11_tmpl.thy
File Edit Search Markers Folding View Utilities Macros Plugins Help
ex11_tmpl.thy (~/lehre/FDS/SS18/public/exercises/)
▪ Show that your function is correct
▪ Define a timing function for your insert function, and show that it is
linearly bounded by the rank of the tree.
›
fun lh_insert :: "'a::ord ⇒ 'a lheap ⇒ 'a lheap"
where
"lh_insert x ⟨⟩ = ⟨1 , ⟨⟩ , x , ⟨⟩ ⟩"
"lh_insert x ⟨_,l,a,r⟩ = (if a<x then node l a (lh_insert x r) else node l x (lh_insert a r))"
lemma mset_lh_insert: "mset_tree (lh_insert x t) = mset_tree t + {# x #}"
sorry
lemma "heap t ⟹ heap (lh_insert x t)"
sorry
lemma "ltree t ⟹ ltree (lh_insert x t)"
sorry
Output Query Sledgehammer Symbols
27.1 (748/5435)
(isabelle,isabelle,UTF-8-Isabelle) UG 1344MB 12:30 PM

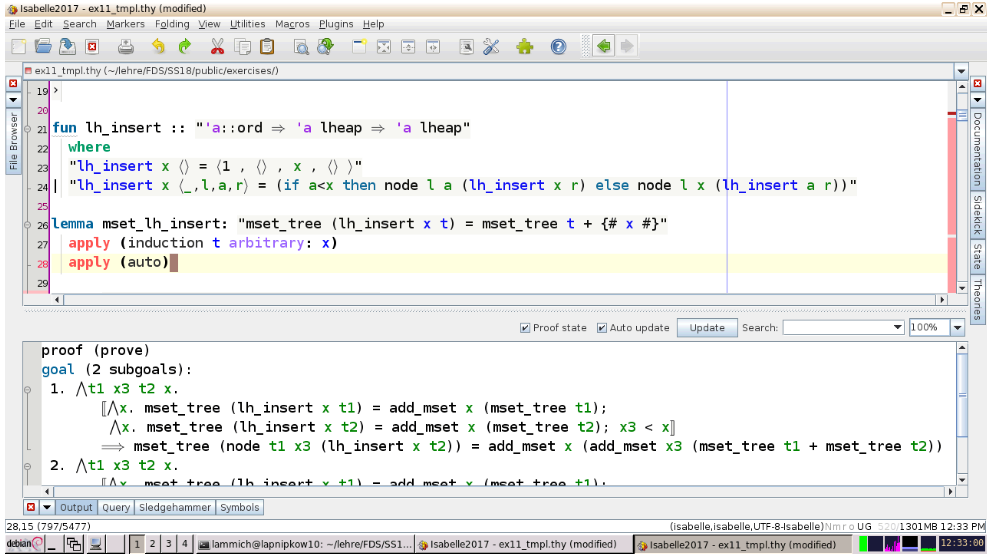

Isabelle2017 - ex11_tmpl.thy (modified)
File Edit Search Markers Folding View Utilities Macros Plugins Help
ex11_tmpl.thy (~/lehre/FDS/SS18/public/exercises/)
▪ Define a timing function for your insert function, and show that it is
linearly bounded by the rank of the tree.
fun lh_insert :: "'a::ord ⇒ 'a lheap ⇒ 'a lheap"
where
"lh_insert x ⟨⟩ = ⟨1 , ⟨⟩ , x , ⟨⟩ ⟩"
| "lh_insert x ⟨_,l,a,r⟩ = (if a<x then node l a (lh_insert x r) else node l x (lh_insert a r))"
lemma mset_lh_insert: "mset_tree (lh_insert x t) = mset_tree t + {# x #}"
s
lemma "heap t ⟹ heap (lh_insert x t)"
sorry
lemma "ltree t ⟹ ltree (lh_insert x t)"
sorry
fun t_lh_insert :: "'a::ord ⇒ 'a lheap ⇒ nat"
Output Query Sledgehammer Symbols
27,4 (751/5431)
(isabelle,isabelle,UTF-8-Isabelle)
344MB 12:30 PM

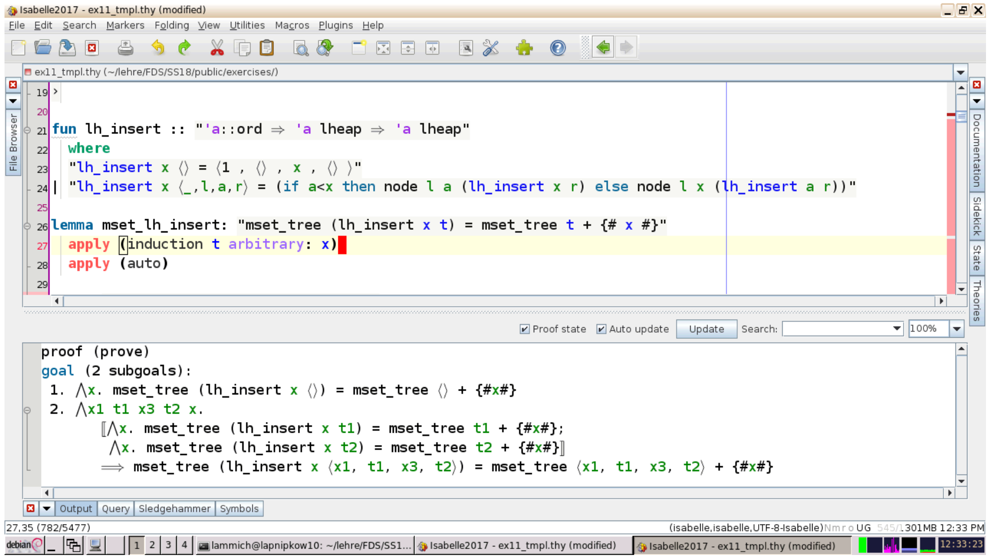

Isabelle2017 - ex11_tmpl.thy (modified)
File Edit Search Markers Folding View Utilities Macros Plugins Help
ex11_tmpl.thy (~/lehre/FDS/SS18/public/exercises/)
fun lh_insert :: "'a::ord ⇒ 'a lheap ⇒ 'a lheap"
where
"lh_insert x ⟨⟩ = ⟨1 , ⟨⟩ , x , ⟨⟩ ⟩"
| "lh_insert x ⟨_,l,a,r⟩ = (if a<x then node l a (lh_insert x r) else node l x (lh_insert a r))"
lemma mset_lh_insert: "mset_tree (lh_insert x t) = mset_tree t + {# x #}"
apply (induction t)
lemma "heap t ⟹ heap (lh_insert x t)"
sorry
lemma "ltree t ⟹ ltree (lh_insert x t)"
Proof state Auto update Update Search: 100%
proof (prove)
goal (2 subgoals):
1. mset_tree (lh_insert x ⟨⟩) = mset_tree ⟨⟩ + {#x#}
2. ⋀x1 t1 x3 t2.
Output Query Sledgehammer Symbols
27,21 (768/5449)
Input/output complete
(isabelle,isabelle,UTF-8-Isabelle)
112/1301MB 12:31 PM

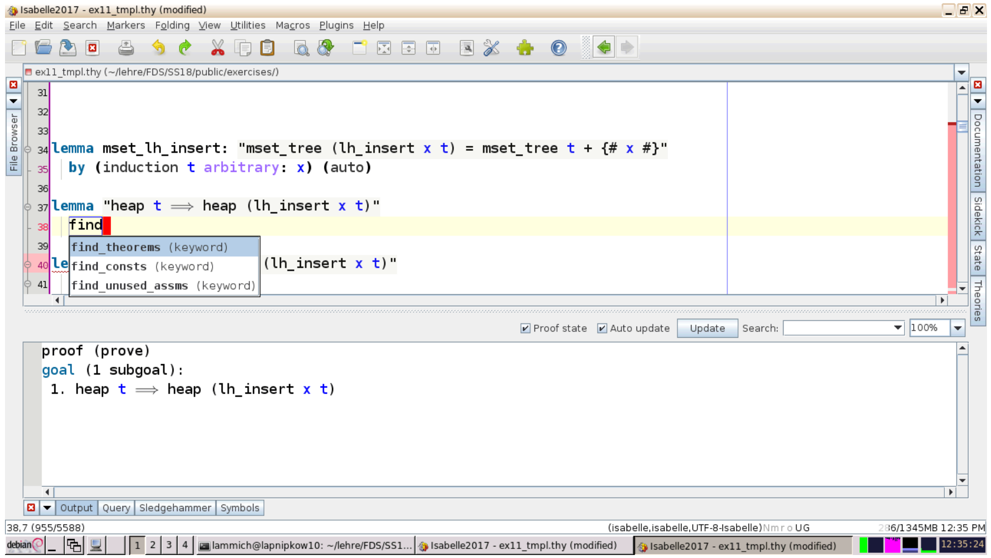

Isabelle2017 - ex11_tmpl.thy (modified)
File Edit Search Markers Folding View Utilities Macros Plugins Help
ex11_tmpl.thy (~/lehre/FDS/SS18/public/exercises/)
fun lh_insert :: "'a::ord ⇒ 'a lheap ⇒ 'a lheap"
where
"lh_insert x ⟨⟩ = ⟨1 , ⟨⟩ , x , ⟨⟩ ⟩"
| "lh_insert x ⟨_,l,a,r⟩ = (if a<x then node l a (lh_insert x r) else node l x (lh_insert a r))"
lemma mset_lh_insert: "mset_tree (lh_insert x t) = mset_tree t + {# x #}"
apply (induction t arb )
lemma "heap t ⟹ heap (lh_insert x t)"
sorry
lemma "ltree t ⟹ ltree (lh_insert x t)"
Proof state Auto update Update Search: 100%
proof (prove)
goal (2 subgoals):
1. mset_tree (lh_insert x ⟨⟩) = mset_tree ⟨⟩ + {#x#}
2. ⋀x1 t1 x3 t2.
Output Query Sledgehammer Symbols
27,26 (773/5454)
(isabelle,isabelle,UTF-8-Isabelle)
118/1301MB 12:31 PM

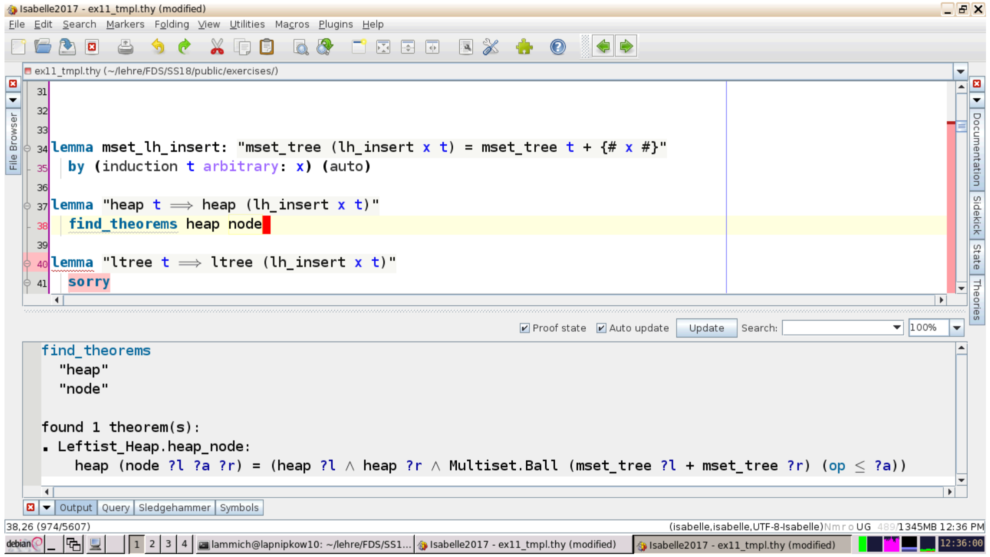

Isabelle2017 - ex11_tmpl.thy (modified)
File Edit Search Markers Folding View Utilities Macros Plugins Help
ex11_tmpl.thy (~/lehre/FDS/SS18/public/exercises/)
fun lh_insert :: "'a::ord ⇒ 'a lheap ⇒ 'a lheap"
where
"lh_insert x ⟨⟩ = ⟨1 , ⟨⟩ , x , ⟨⟩ ⟩"
| "lh_insert x ⟨_,l,a,r⟩ = (if a<x then node l a (lh_insert x r) else node l x (lh_insert a r))"
lemma mset_lh_insert: "mset_tree (lh_insert x t) = mset_tree t + {# x #}"
apply (induction t arbitrary: x)
apply (auto)
Proof state Auto update Update Search: 100%
proof (prove)
goal (2 subgoals):
1. ⋀t1 x3 t2 x.
⟦⋀x. mset_tree (lh_insert x t1) = add_mset x (mset_tree t1);
⋀x. mset_tree (lh_insert x t2) = add_mset x (mset_tree t2); x3 < x⟧
⟹ mset_tree (node t1 x3 (lh_insert x t2)) = add_mset x (add_mset x3 (mset_tree t1 + mset_tree t2))
2. ⋀t1 x3 t2 x.
Output Query Sledgehammer Symbols
28,15 (797/5477)
Input/output complete
(isabelle,isabelle,UTF-8-Isabelle)
1301MB 12:32 PM


Isabelle2017 - ex11_tmpl.thy (modified)
fun lh_insert :: "'a::ord ⇒ 'a lheap ⇒ 'a lheap"
where
"lh_insert x ⟨⟩ = ⟨1 , ⟨⟩ , x , ⟨⟩ ⟩"
| "lh_insert x ⟨_,l,a,r⟩ = (if a<x then node l a (lh_insert x r) else node l x (lh_insert a r))"
lemma mset_lh_insert: "mset_tree (lh_insert x t) = mset_tree t + {# x #}"
apply (induction t arbitrary: x)
apply (auto)
proof (prove)
goal (2 subgoals):
1. ⋀t1 x3 t2 x.
⟦⋀x. mset_tree (lh_insert x t1) = add_mset x (mset_tree t1);
⋀x. mset_tree (lh_insert x t2) = add_mset x (mset_tree t2); x3 < x⟧
⟹ mset_tree (node t1 x3 (lh_insert x t2)) = add_mset x (add_mset x3 (mset_tree t1 + mset_tree t2))
2. ⋀t1 x3 t2 x.

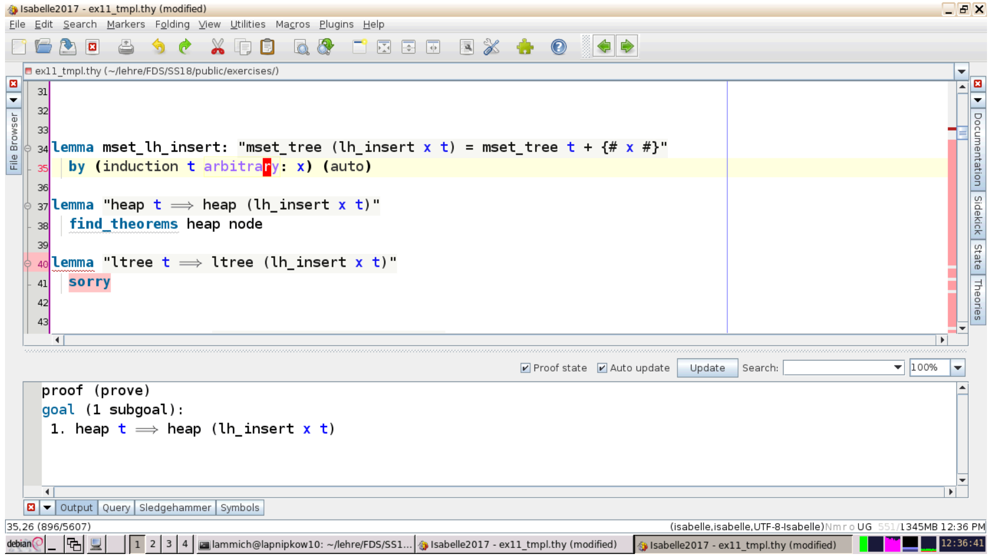

Isabelle2017 - ex11_tmpl.thy (modified)
fun lh_insert :: "'a::ord ⇒ 'a lheap ⇒ 'a lheap"
where
"lh_insert x ⟨⟩ = ⟨1 , ⟨⟩ , x , ⟨⟩ ⟩"
| "lh_insert x ⟨_,l,a,r⟩ = (if a<x then node l a (lh_insert x r) else node l x (lh_insert a r))"
lemma mset_lh_insert: "mset_tree (lh_insert x t) = mset_tree t + {# x #}"
apply (induction t arbitrary: x)
apply (auto)
proof (prove)
goal (2 subgoals):
1. ⋀t1 x3 t2 x.
⟦⋀x. mset_tree (lh_insert x t1) = add_mset x (mset_tree t1);
⋀x. mset_tree (lh_insert x t2) = add_mset x (mset_tree t2); x3 < x⟧
⟹ mset_tree (node t1 x3 (lh_insert x t2)) = add_mset x (add_mset x3 (mset_tree t1 + mset_tree t2))
2. ⋀t1 x3 t2 x.


Isabelle2017 - ex11_tmpl.thy (modified)
fun lh_insert :: "'a::ord ⇒ 'a lheap ⇒ 'a lheap"
where
"lh_insert x ⟨⟩ = ⟨1 , ⟨⟩ , x , ⟨⟩ ⟩"
| "lh_insert x ⟨_,l,a,r⟩ = (if a<x then node l a (lh_insert x r) else node l x (lh_insert a r))"
lemma mset_lh_insert: "mset_tree (lh_insert x t) = mset_tree t + {# x #}"
apply (induction t arbitrary: x)
apply (auto)
proof (prove)
goal (2 subgoals):
1. ⋀x. mset_tree (lh_insert x ⟨⟩) = mset_tree ⟨⟩ + {#x#}
2. ⋀x1 t1 x3 t2 x.
⟦⋀x. mset_tree (lh_insert x t1) = mset_tree t1 + {#x#};
⋀x. mset_tree (lh_insert x t2) = mset_tree t2 + {#x#}⟧
⟹ mset_tree (lh_insert x ⟨x1, t1, x3, t2⟩) = mset_tree ⟨x1, t1, x3, t2⟩ + {#x#}

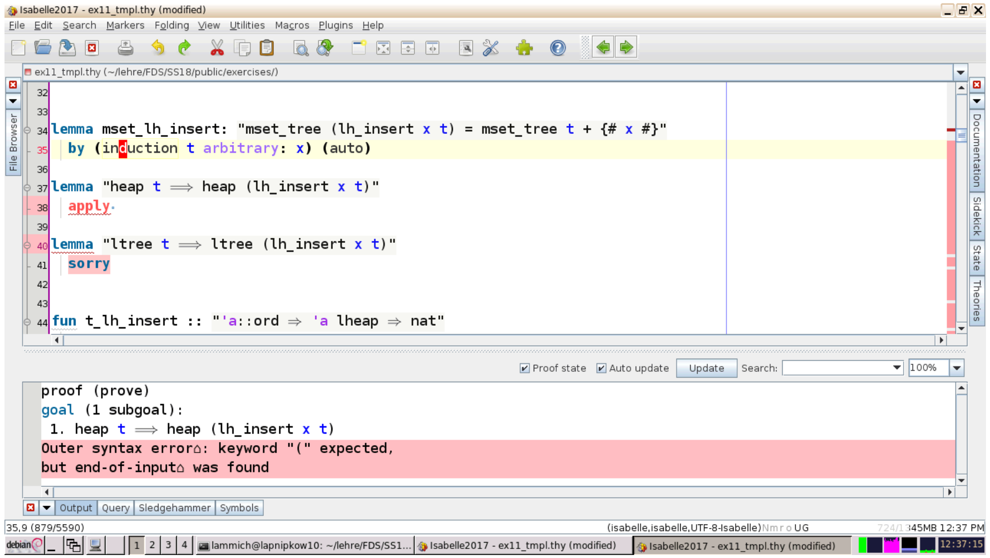

Isabelle2017 - ex11_tmpl.thy (modified)
lemma mset_lh_insert: "mset_tree (lh_insert x t) = mset_tree t + {# x #}"
by (induction t arbitrary: x) (auto)
lemma "heap t ⟹ heap (lh_insert x t)"
find
find_theorems (keyword)
find_consts (keyword)
find_unused_assms (keyword)
proof (prove)
goal (1 subgoal):
1. heap t ⟹ heap (lh_insert x t)



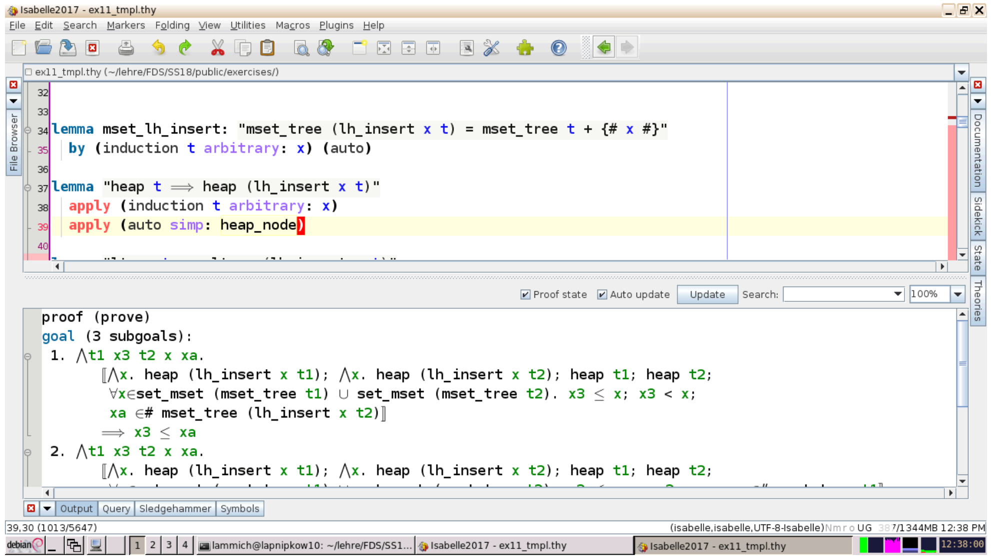
Isabelle2017 - ex11_tmpl.thy (modified)
lemma mset_lh_insert: "mset_tree (lh_insert x t) = mset_tree t + {# x #}"
by (induction t arbitrary: x) (auto)
lemma "heap t ⟹ heap (lh_insert x t)"
find_theorems heap node
lemma "ltree t ⟹ ltree (lh_insert x t)"
sorry
find_theorems
"heap"
"node"
found 1 theorem(s):
• Leftist_Heap.heap_node:
heap (node ?l ?a ?r) = (heap ?l ∧ heap ?r ∧ Multiset.Ball (mset_tree ?l + mset_tree ?r) (op ≤ ?a))

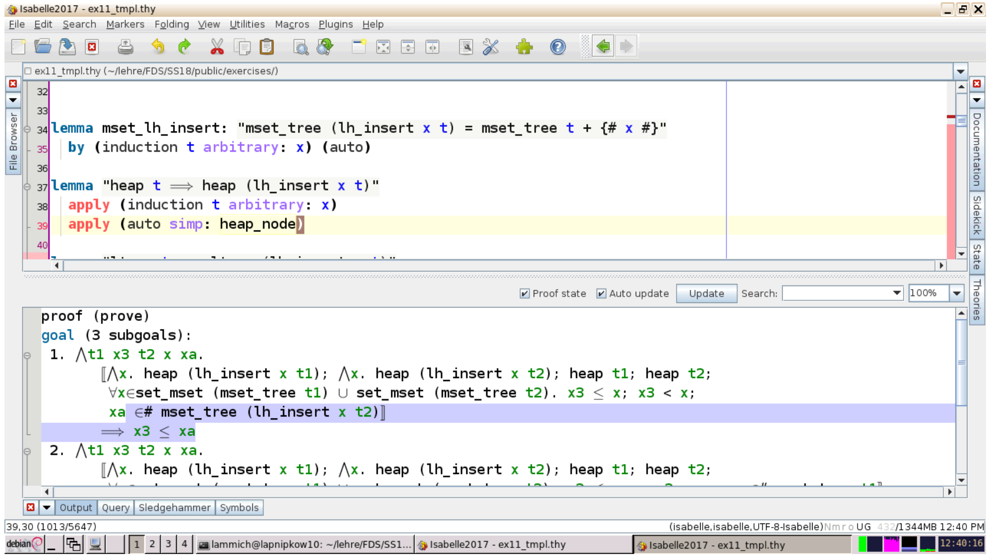
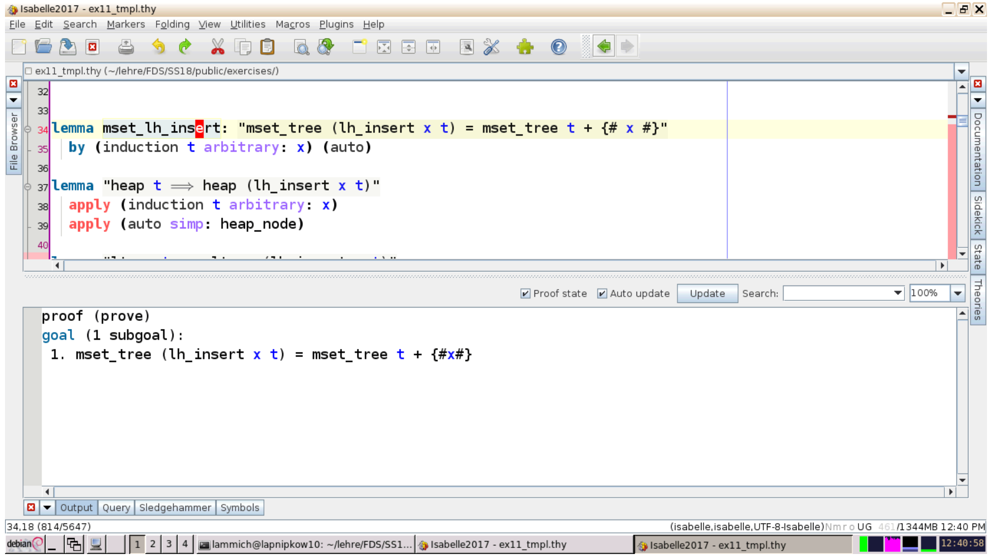
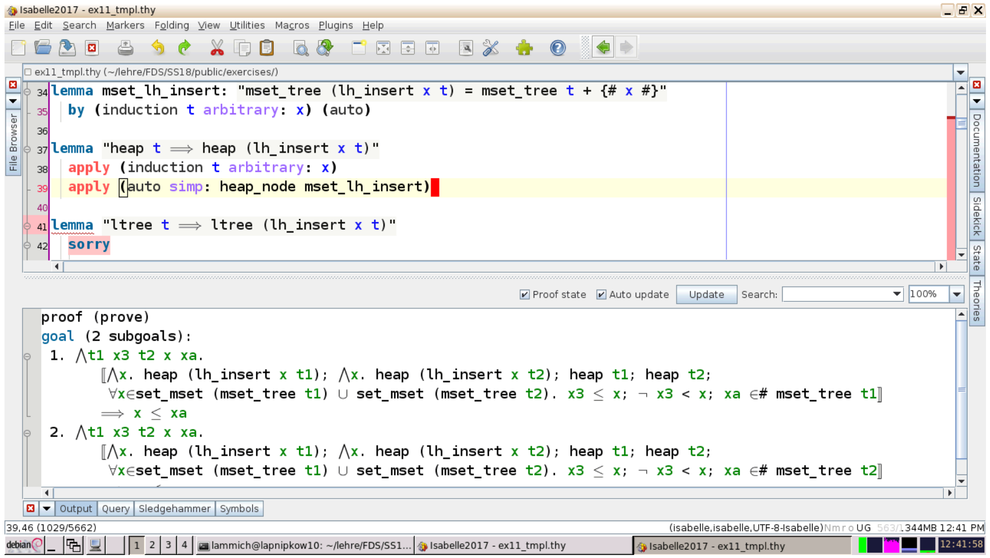
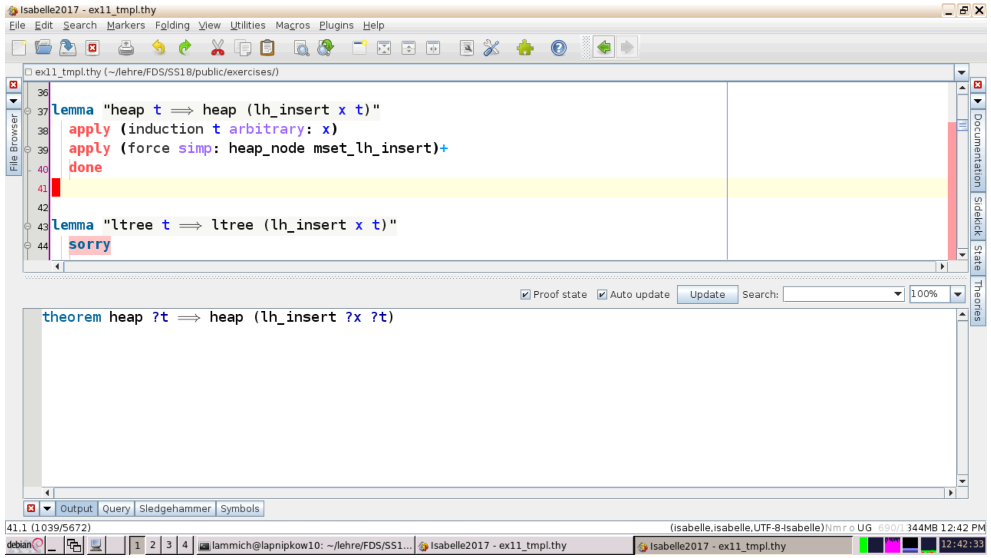
Isabelle2017 - ex11_tmpl.thy (modified)
lemma mset_lh_insert: "mset_tree (lh_insert x t) = mset_tree t + {# x #}"
by (induction t arbitrary: x) (auto)
lemma "heap t ⟹ heap (lh_insert x t)"
find_theorems heap node
lemma "ltree t ⟹ ltree (lh_insert x t)"
sorry
proof (prove)
goal (1 subgoal):
1. heap t ⟹ heap (lh_insert x t)

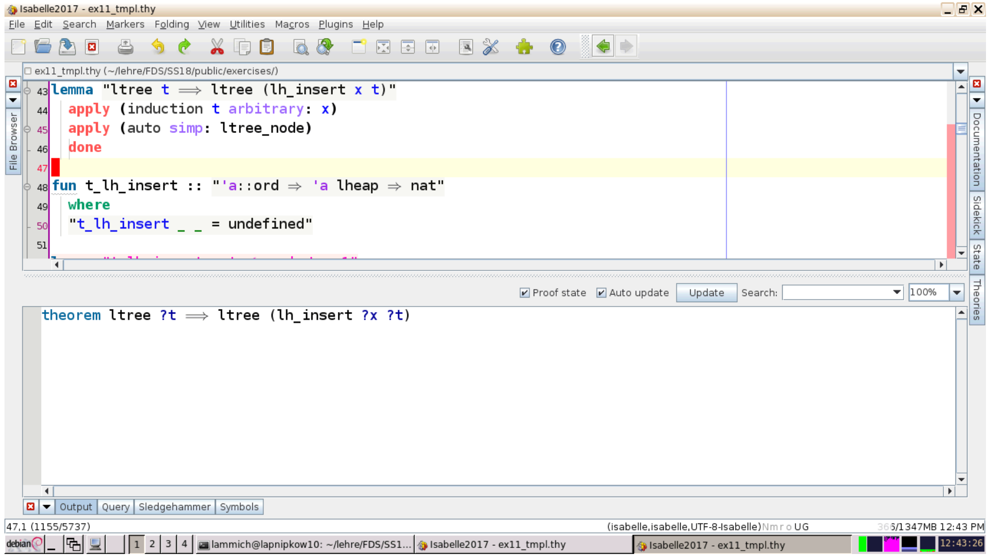
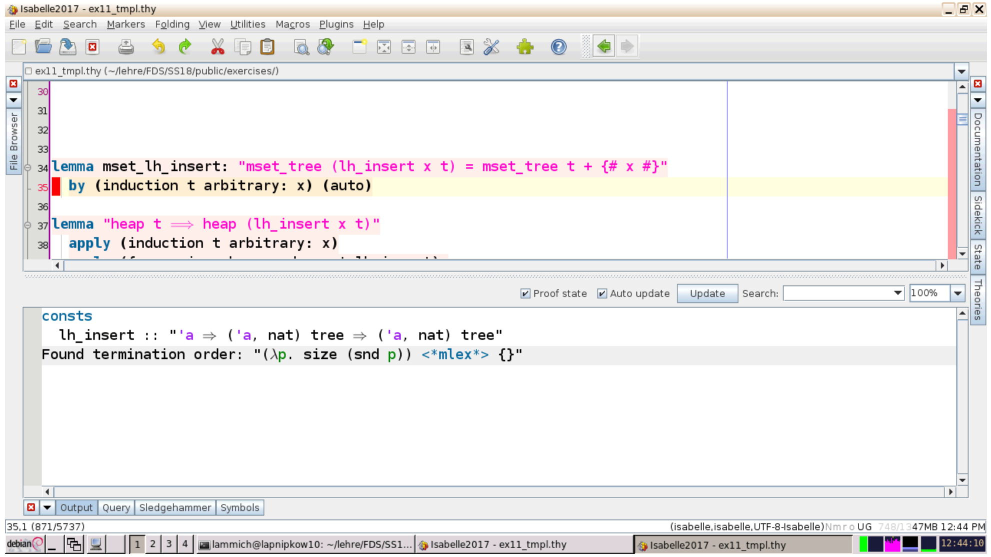
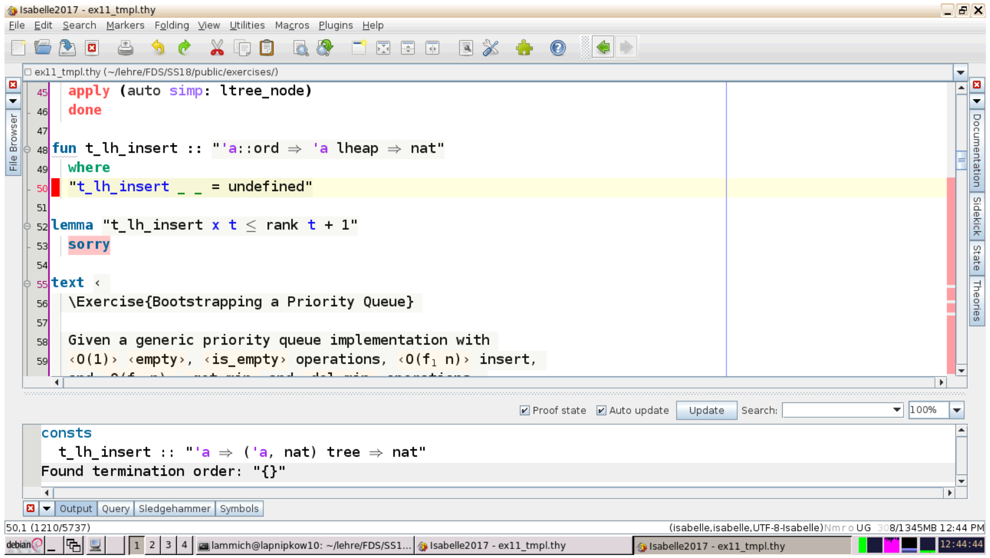
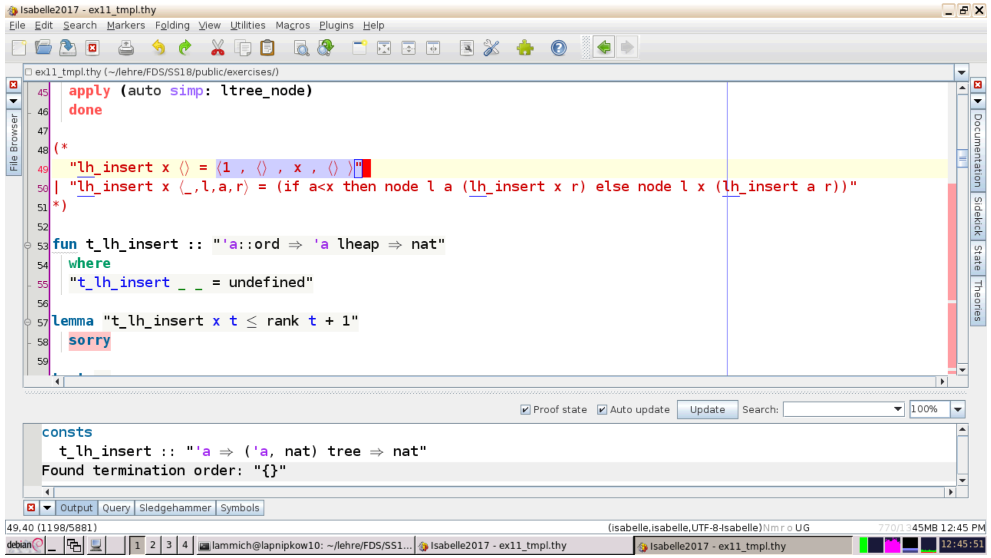



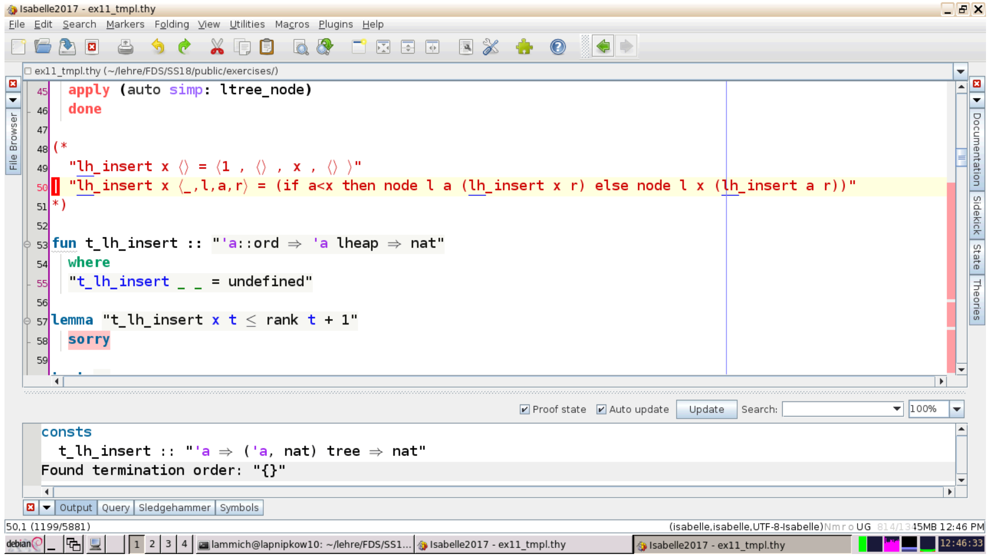
Isabelle2017 - ex11_tmpl.thy (modified)
lemma mset_lh_insert: "mset_tree (lh_insert x t) = mset_tree t + {# x #}"
by (induction t arbitrary: x) (auto)
lemma "heap t ⟹ heap (lh_insert x t)"
apply
lemma "ltree t ⟹ ltree (lh_insert x t)"
sorry
fun t_lh_insert :: "'a::ord ⇒ 'a lheap ⇒ nat"
proof (prove)
goal (1 subgoal):
1. heap t ⟹ heap (lh_insert x t)
Outer syntax error: keyword "(" expected,
but end-of-input was found

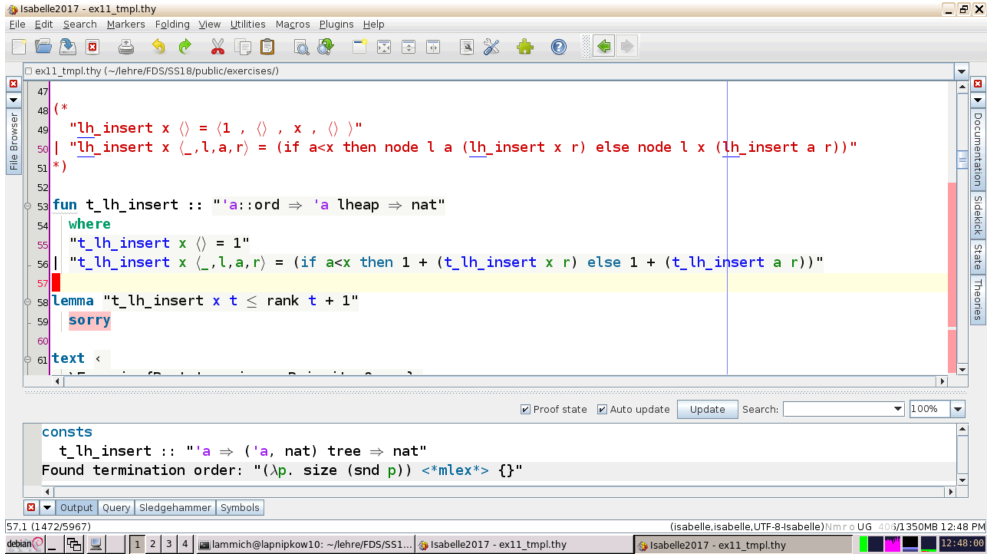



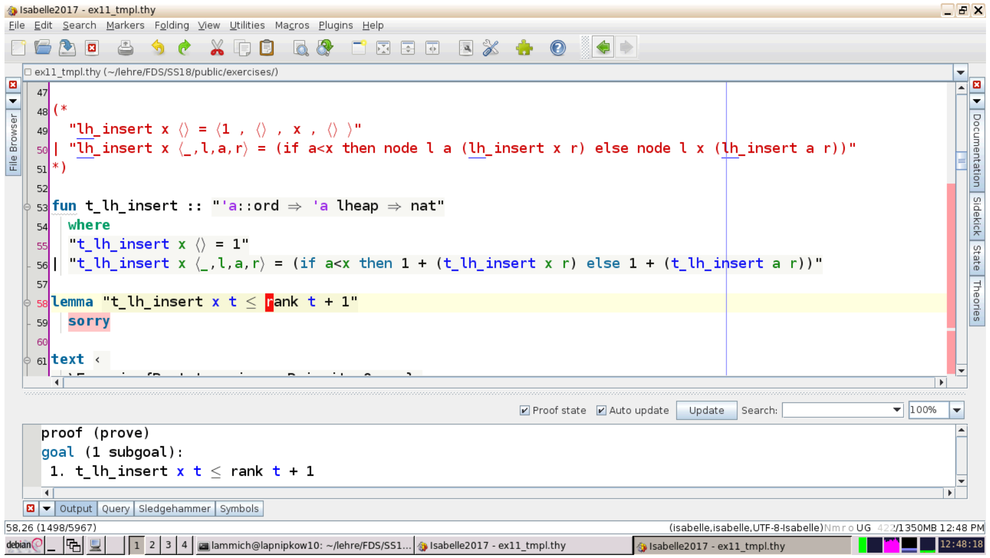
Isabelle2017 - ex11_tmpl.thy
lemma mset_lh_insert: "mset_tree (lh_insert x t) = mset_tree t + {# x #}"
by (induction t arbitrary: x) (auto)
lemma "heap t ⟹ heap (lh_insert x t)"
apply (induction t arbitrary: x)
apply (auto simp: heap_node)
proof (prove)
goal (3 subgoals):
1. ⋀t1 x3 t2 x xa.
⟦⋀x. heap (lh_insert x t1); ⋀x. heap (lh_insert x t2); heap t1; heap t2;
∀x∈set_mset (mset_tree t1) ∪ set_mset (mset_tree t2). x3 ≤ x; x3 < x;
xa ∈# mset_tree (lh_insert x t2)⟧
⟹ x3 ≤ xa
2. ⋀t1 x3 t2 x xa.
⟦⋀x. heap (lh_insert x t1); ⋀x. heap (lh_insert x t2); heap t1; heap t2;

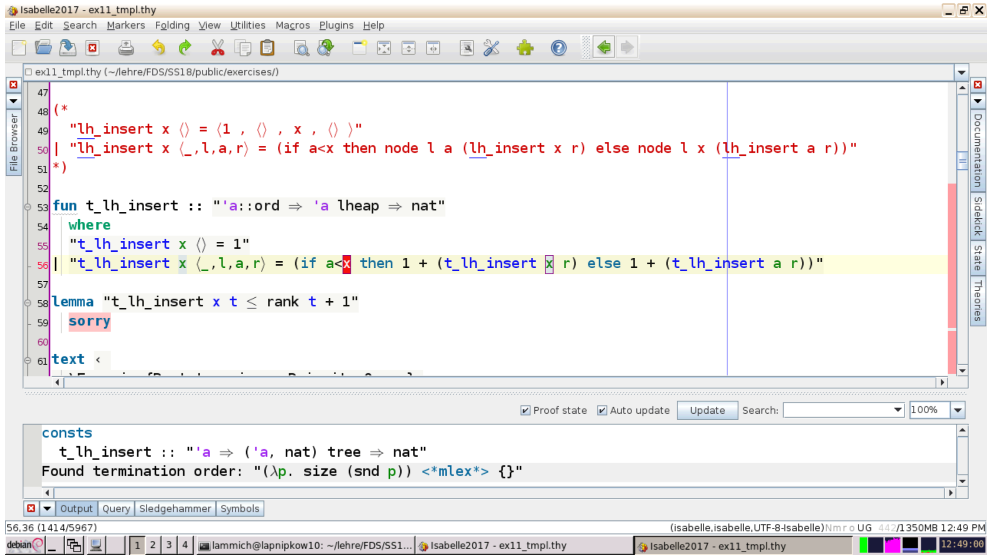

Isabelle2017 - ex11_tmpl.thy
lemma mset_lh_insert: "mset_tree (lh_insert x t) = mset_tree t + {# x #}"
by (induction t arbitrary: x) (auto)
lemma "heap t ⟹ heap (lh_insert x t)"
apply (induction t arbitrary: x)
apply (auto simp: heap_node)
proof (prove)
goal (3 subgoals):
1. ⋀t1 x3 t2 x xa.
⟦⋀x. heap (lh_insert x t1); ⋀x. heap (lh_insert x t2); heap t1; heap t2;
∀x∈set_mset (mset_tree t1) ∪ set_mset (mset_tree t2). x3 ≤ x; x3 < x;
xa ∈# mset_tree (lh_insert x t2)⟧
⟹ x3 ≤ xa
2. ⋀t1 x3 t2 x xa.
⟦⋀x. heap (lh_insert x t1); ⋀x. heap (lh_insert x t2); heap t1; heap t2;

Isabelle2017 - ex11_tmpl.thy
lemma mset_lh_insert: "mset_tree (lh_insert x t) = mset_tree t + {# x #}"
by (induction t arbitrary: x) (auto)
lemma "heap t ⟹ heap (lh_insert x t)"
apply (induction t arbitrary: x)
apply (auto simp: heap_node)
proof (prove)
goal (1 subgoal):
1. mset_tree (lh_insert x t) = mset_tree t + {#x#}

Isabelle2017 - ex11_tmpl.thy
lemma mset_lh_insert: "mset_tree (lh_insert x t) = mset_tree t + {# x #}"
by (induction t arbitrary: x) (auto)
lemma "heap t ⟹ heap (lh_insert x t)"
apply (induction t arbitrary: x)
apply (auto simp: heap_node mset_lh_insert)
lemma "ltree t ⟹ ltree (lh_insert x t)"
sorry
proof (prove)
goal (2 subgoals):
1. ⋀t1 x3 t2 x xa.
⟦⋀x. heap (lh_insert x t1); ⋀x. heap (lh_insert x t2); heap t1; heap t2;
∀x∈set_mset (mset_tree t1) ∪ set_mset (mset_tree t2). x3 ≤ x; ¬ x3 < x; xa ∈# mset_tree t1⟧
⟹ x ≤ xa
2. ⋀t1 x3 t2 x xa.
⟦⋀x. heap (lh_insert x t1); ⋀x. heap (lh_insert x t2); heap t1; heap t2;
∀x∈set_mset (mset_tree t1) ∪ set_mset (mset_tree t2). x3 ≤ x; ¬ x3 < x; xa ∈# mset_tree t2⟧

Isabelle2017 - ex11_tmpl.thy
lemma "heap t ⟹ heap (lh_insert x t)"
apply (induction t arbitrary: x)
apply (force simp: heap_node mset_lh_insert)+
done
lemma "ltree t ⟹ ltree (lh_insert x t)"
sorry
theorem heap ?t ⟹ heap (lh_insert ?x ?t)

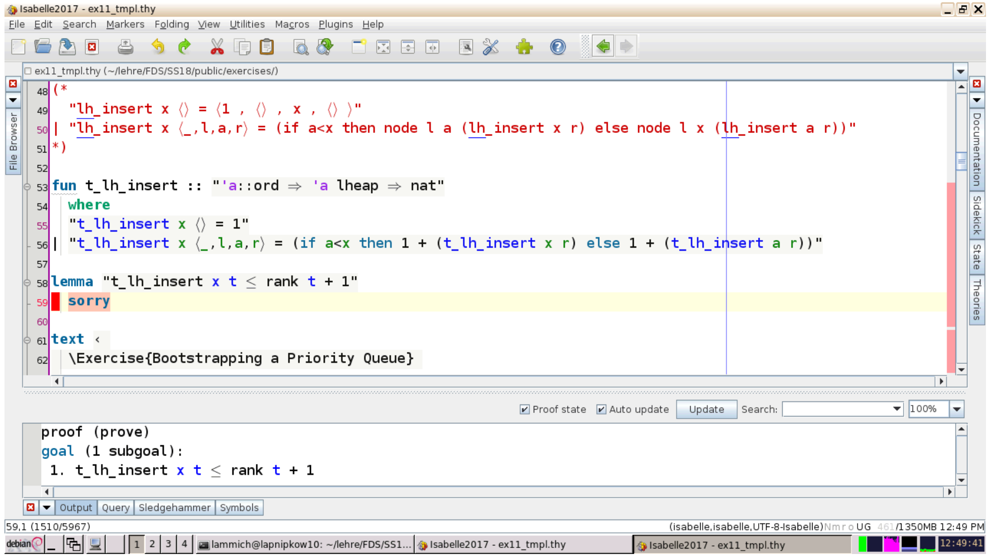

Isabelle2017 - ex11_tmpl.thy
lemma "ltree t ⟹ ltree (lh_insert x t)"
apply (induction t arbitrary: x)
apply (auto simp: ltree_node)
done
fun t_lh_insert :: "'a::ord ⇒ 'a lheap ⇒ nat"
where
"t_lh_insert _ _ = undefined"
Proof state
Auto update
Update
Search:
100%
theorem ltree ?t ⟹ ltree (lh_insert ?x ?t)
Output
Query
Sledgehammer
Symbols
47,1 (1155/5737)
(isabelle,isabelle,UTF-8-Isabelle)
12:43 PM

Isabelle2017 - ex11_tmpl.thy
lemma mset_lh_insert: "mset_tree (lh_insert x t) = mset_tree t + {# x #}"
by (induction t arbitrary: x) (auto)
lemma "heap t ⟹ heap (lh_insert x t)"
apply (induction t arbitrary: x)
Proof state
Auto update
Update
Search:
100%
consts
lh_insert :: "'a ⇒ ('a, nat) tree ⇒ ('a, nat) tree"
Found termination order: "(λp. size (snd p)) <*mlex*> {}"
Output
Query
Sledgehammer
Symbols
35,1 (871/5737)
(isabelle,isabelle,UTF-8-Isabelle)
12:44 PM

Isabelle2017 - ex11_tmpl.thy
apply (auto simp: ltree_node)
done
fun t_lh_insert :: "'a::ord ⇒ 'a lheap ⇒ nat"
where
"t_lh_insert _ _ = undefined"
lemma "t_lh_insert x t ≤ rank t + 1"
sorry
text ‹
\Exercise{Bootstrapping a Priority Queue}
Given a generic priority queue implementation with
‹O(1)› ‹empty›, ‹is_empty› operations, ‹O(f₁ n)› insert,
Proof state
Auto update
Update
Search:
100%
consts
t_lh_insert :: "'a ⇒ ('a, nat) tree ⇒ nat"
Found termination order: "{}"
Output
Query
Sledgehammer
Symbols
50,1 (1210/5737)
(isabelle,isabelle,UTF-8-Isabelle)
12:44 PM

Isabelle2017 - ex11_tmpl.thy
apply (auto simp: ltree_node)
done
(*
"lh_insert x ⟨⟩ = ⟨1 , ⟨⟩ , x , ⟨⟩ ⟩"
| "lh_insert x ⟨_,l,a,r⟩ = (if a<x then node l a (lh_insert x r) else node l x (lh_insert a r))"
*)
fun t_lh_insert :: "'a::ord ⇒ 'a lheap ⇒ nat"
where
"t_lh_insert _ _ = undefined"
lemma "t_lh_insert x t ≤ rank t + 1"
sorry
Proof state
Auto update
Update
Search:
100%
consts
t_lh_insert :: "'a ⇒ ('a, nat) tree ⇒ nat"
Found termination order: "{}"
Output
Query
Sledgehammer
Symbols
49,40 (1198/5881)
(isabelle,isabelle,UTF-8-Isabelle)
12:45 PM

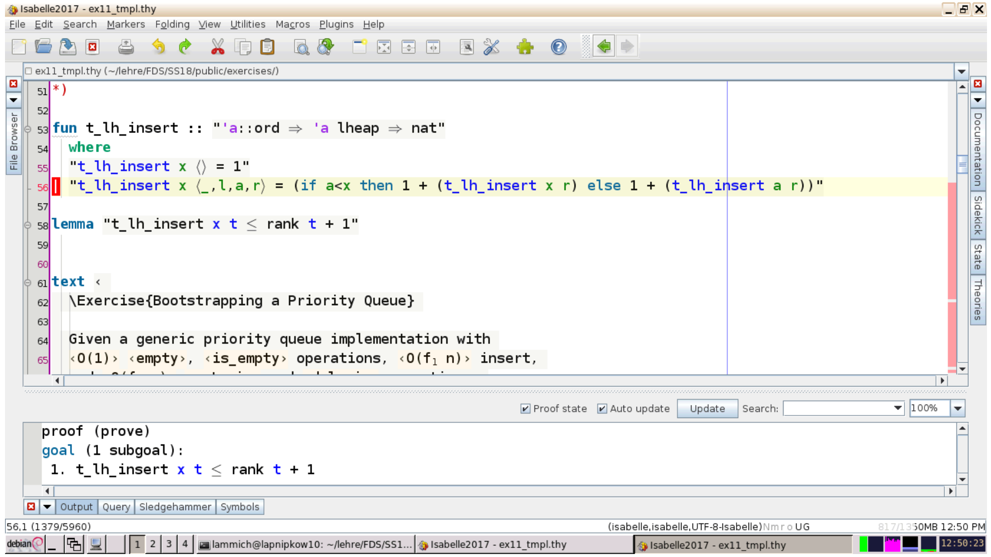

Isabelle2017 - ex11_tmpl.thy
ex11_tmpl.thy (~/lehre/FDS/SS18/public/exercises/)
apply (auto simp: ltree_node)
done
(*
"lh_insert x ⟨⟩ = ⟨1 , ⟨⟩ , x , ⟨⟩ ⟩"
"lh_insert x ⟨_,l,a,r⟩ = (if a<x then node l a (lh_insert x r) else node l x (lh_insert a r))"
*)
fun t_lh_insert :: "'a::ord ⇒ 'a lheap ⇒ nat"
where
"t_lh_insert _ _ = undefined"
lemma "t_lh_insert x t ≤ rank t + 1"
sorry
consts
t_lh_insert :: "'a ⇒ ('a, nat) tree ⇒ nat"
Found termination order: "{}"

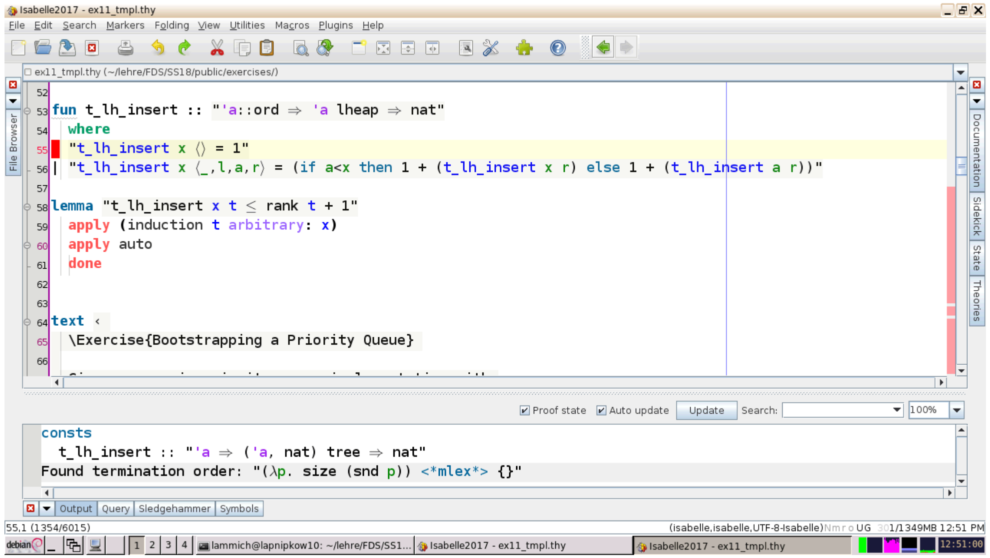

Isabelle2017 - ex11_tmpl.thy
ex11_tmpl.thy (~/lehre/FDS/SS18/public/exercises/)
(*
"lh_insert x ⟨⟩ = ⟨1 , ⟨⟩ , x , ⟨⟩ ⟩"
"lh_insert x ⟨_,l,a,r⟩ = (if a<x then node l a (lh_insert x r) else node l x (lh_insert a r))"
*)
fun t_lh_insert :: "'a::ord ⇒ 'a lheap ⇒ nat"
where
"t_lh_insert x ⟨⟩ = 1"
"t_lh_insert x ⟨_,l,a,r⟩ = (if a<x then 1 + (t_lh_insert x r) else 1 + (t_lh_insert a r))"
lemma "t_lh_insert x t ≤ rank t + 1"
sorry
text ‹
consts
t_lh_insert :: "'a ⇒ ('a, nat) tree ⇒ nat"
Found termination order: "(λp. size (snd p)) <*mlex*> {}"

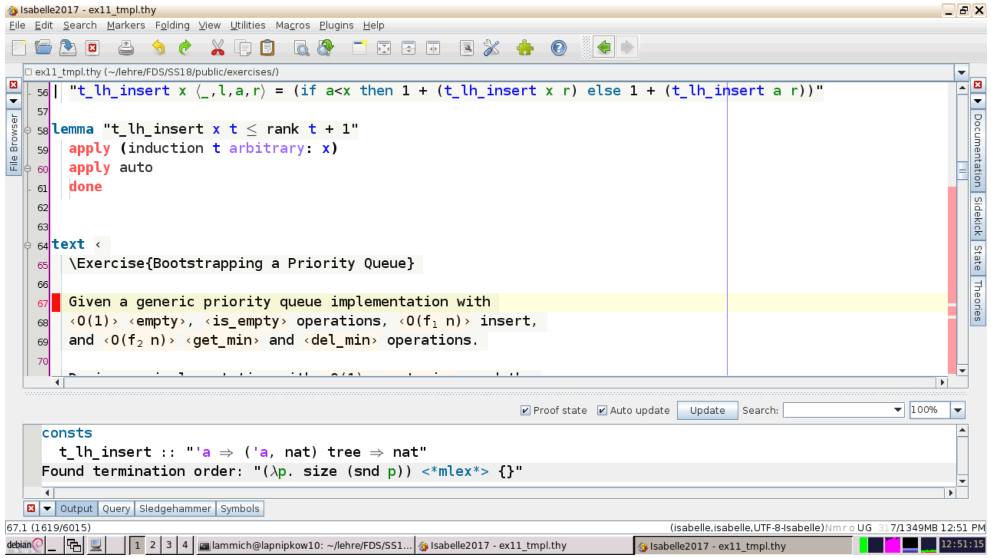

Isabelle2017 - ex11_tmpl.thy
ex11_tmpl.thy (~/lehre/FDS/SS18/public/exercises/)
(*
"lh_insert x ⟨⟩ = ⟨1 , ⟨⟩ , x , ⟨⟩ ⟩"
"lh_insert x ⟨_,l,a,r⟩ = (if a<x then node l a (lh_insert x r) else node l x (lh_insert a r))"
*)
fun t_lh_insert :: "'a::ord ⇒ 'a lheap ⇒ nat"
where
"t_lh_insert x ⟨⟩ = 1"
"t_lh_insert x ⟨_,l,a,r⟩ = (if a<x then 1 + (t_lh_insert x r) else 1 + (t_lh_insert a r))"
lemma "t_lh_insert x t ≤ rank t + 1"
sorry
text ‹
proof (prove)
goal (1 subgoal):
1. t_lh_insert x t ≤ rank t + 1

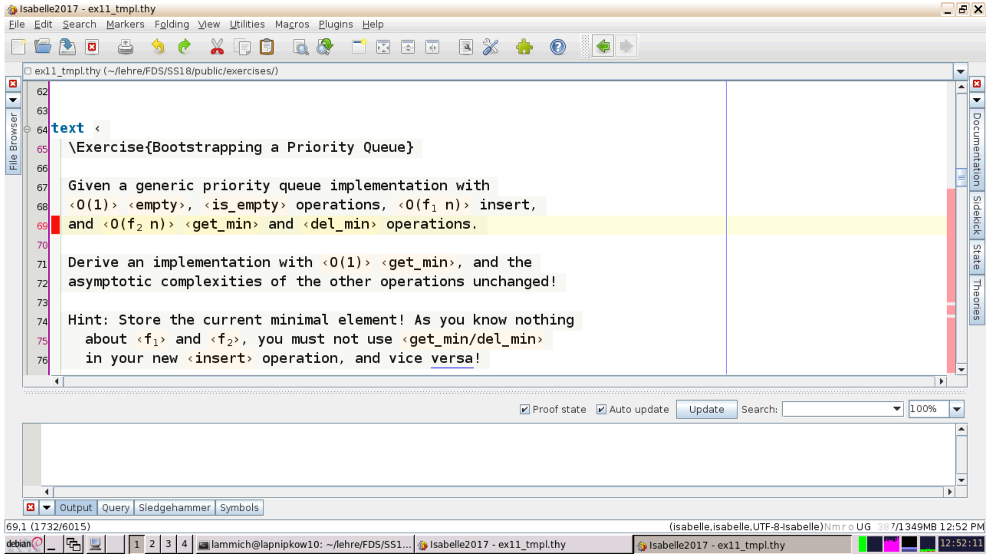

Isabelle2017 - ex11_tmpl.thy
ex11_tmpl.thy (~/lehre/FDS/SS18/public/exercises/)
(*
"lh_insert x ⟨⟩ = ⟨1 , ⟨⟩ , x , ⟨⟩ ⟩"
"lh_insert x ⟨_,l,a,r⟩ = (if a<x then node l a (lh_insert x r) else node l x (lh_insert a r))"
*)
fun t_lh_insert :: "'a::ord ⇒ 'a lheap ⇒ nat"
where
"t_lh_insert x ⟨⟩ = 1"
"t_lh_insert x ⟨_,l,a,r⟩ = (if a<x then 1 + (t_lh_insert x r) else 1 + (t_lh_insert a r))"
lemma "t_lh_insert x t ≤ rank t + 1"
sorry
text ‹
consts
t_lh_insert :: "'a ⇒ ('a, nat) tree ⇒ nat"
Found termination order: "(λp. size (snd p)) <*mlex*> {}"

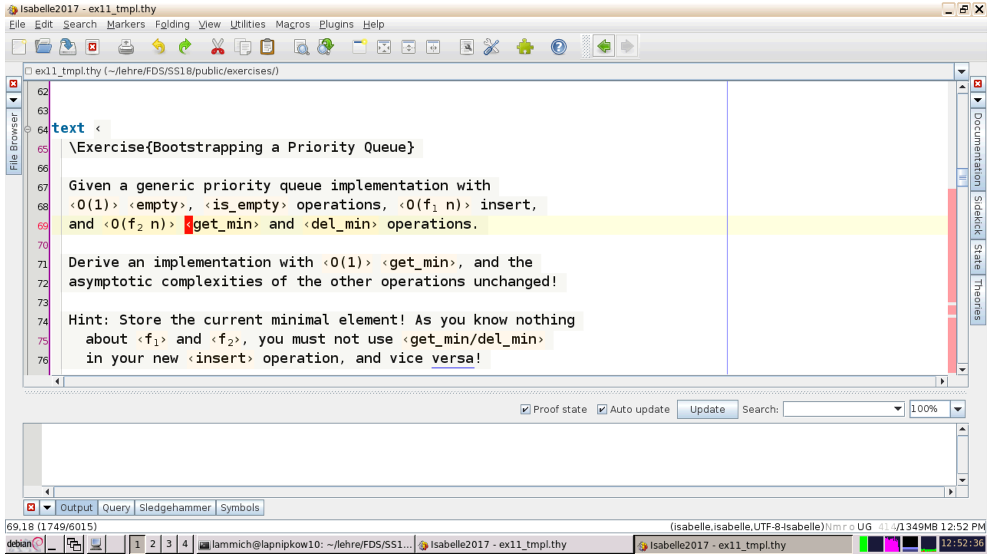

Isabelle2017 - ex11_tmpl.thy
(*
"lh_insert x ⟨⟩ = ⟨1 , ⟨⟩ , x , ⟨⟩ ⟩"
| "lh_insert x ⟨_,l,a,r⟩ = (if a<x then node l a (lh_insert x r) else node l x (lh_insert a r))"
*)
fun t_lh_insert :: "'a::ord ⇒ 'a lheap ⇒ nat"
where
"t_lh_insert x ⟨⟩ = 1"
| "t_lh_insert x ⟨_,l,a,r⟩ = (if a<x then 1 + (t_lh_insert x r) else 1 + (t_lh_insert a r))"
lemma "t_lh_insert x t ≤ rank t + 1"
sorry
text ‹
\Exercise{Bootstrapping a Priority Queue}
proof (prove)
goal (1 subgoal):
1. t_lh_insert x t ≤ rank t + 1

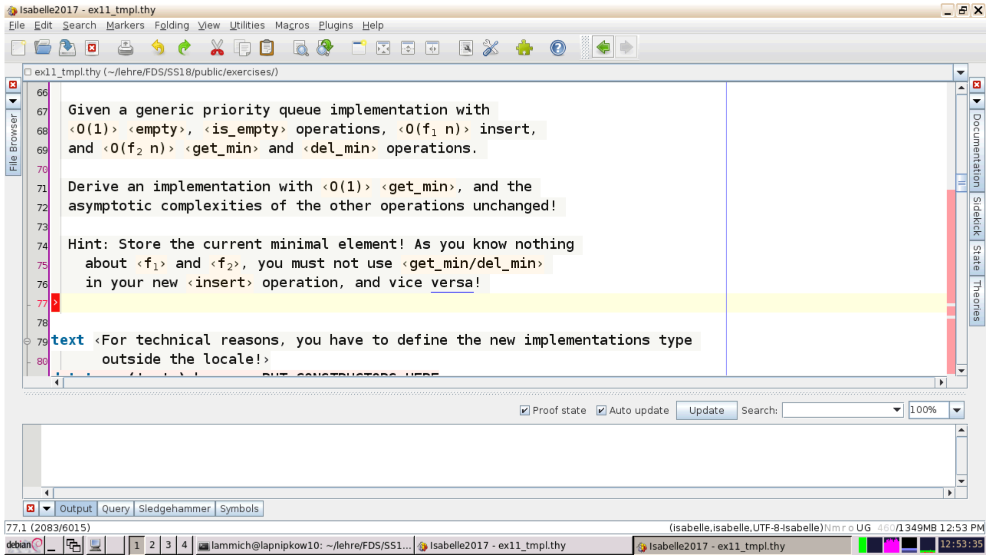

Isabelle2017 - ex11_tmpl.thy
*)
fun t_lh_insert :: "'a::ord ⇒ 'a lheap ⇒ nat"
where
"t_lh_insert x ⟨⟩ = 1"
"t_lh_insert x ⟨_,l,a,r⟩ = (if a<x then 1 + (t_lh_insert x r) else 1 + (t_lh_insert a r))"
lemma "t_lh_insert x t ≤ rank t + 1"
text ‹
\Exercise{Bootstrapping a Priority Queue}
Given a generic priority queue implementation with
‹O(1)› ‹empty›, ‹is_empty› operations, ‹O(f₁ n)› insert,
proof (prove)
goal (1 subgoal):
1. t_lh_insert x t ≤ rank t + 1

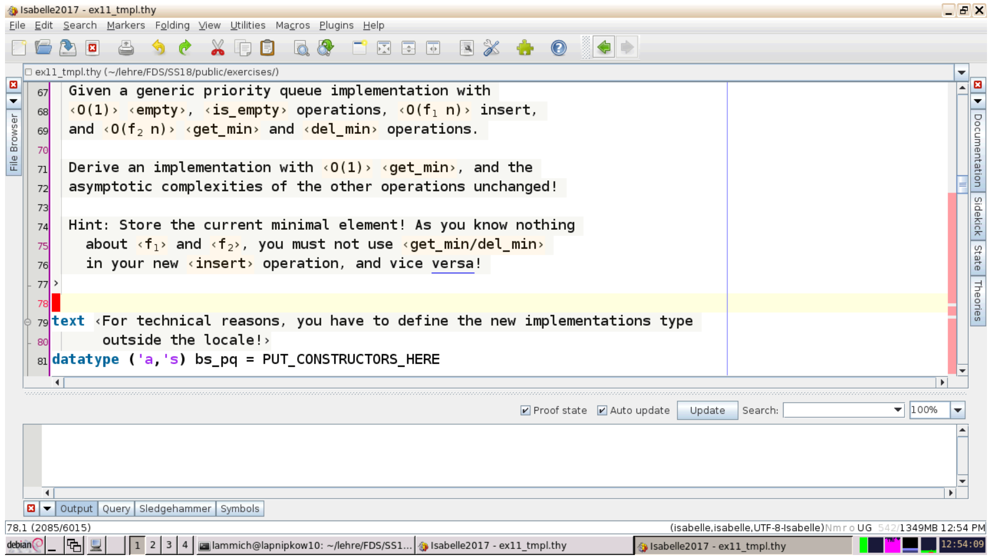

Isabelle2017 - ex11_tmpl.thy
fun t_lh_insert :: "'a::ord ⇒ 'a lheap ⇒ nat"
where
"t_lh_insert x ⟨⟩ = 1"
| "t_lh_insert x ⟨_,l,a,r⟩ = (if a<x then 1 + (t_lh_insert x r) else 1 + (t_lh_insert a r))"
lemma "t_lh_insert x t ≤ rank t + 1"
apply (induction t arbitrary: x)
apply auto
done
text ‹
\Exercise{Bootstrapping a Priority Queue}
consts
t_lh_insert :: "'a ⇒ ('a, nat) tree ⇒ nat"
Found termination order: "(λp. size (snd p)) <*mlex*> {}"

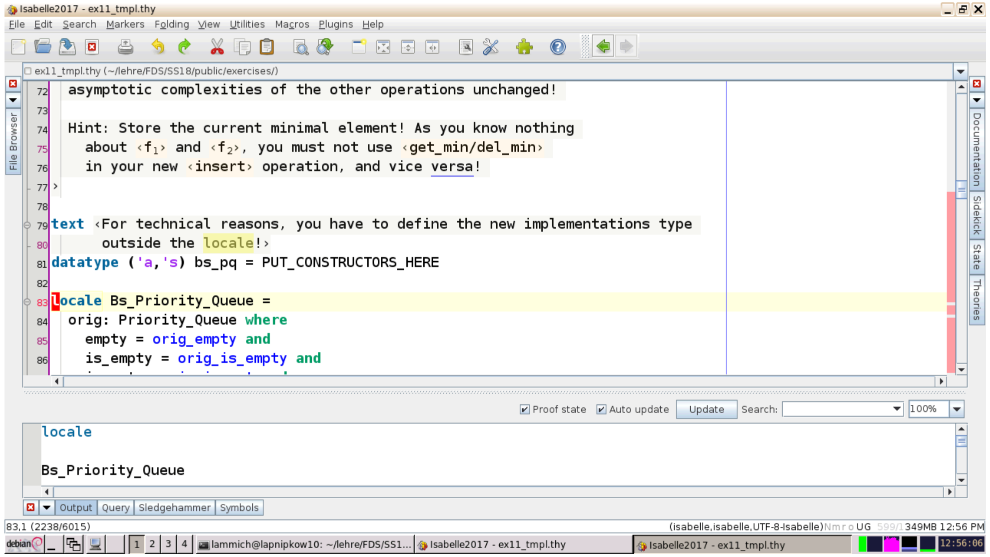

Isabelle2017 - ex11_tmpl.thy
| "t_lh_insert x ⟨_,l,a,r⟩ = (if a<x then 1 + (t_lh_insert x r) else 1 + (t_lh_insert a r))"
lemma "t_lh_insert x t ≤ rank t + 1"
apply (induction t arbitrary: x)
apply auto
done
text ‹
\Exercise{Bootstrapping a Priority Queue}
Given a generic priority queue implementation with
‹O(1)› ‹empty›, ‹is_empty› operations, ‹O(f₁ n)› insert,
and ‹O(f₂ n)› ‹get_min› and ‹del_min› operations.
consts
t_lh_insert :: "'a ⇒ ('a, nat) tree ⇒ nat"
Found termination order: "(λp. size (snd p)) <*mlex*> {}"

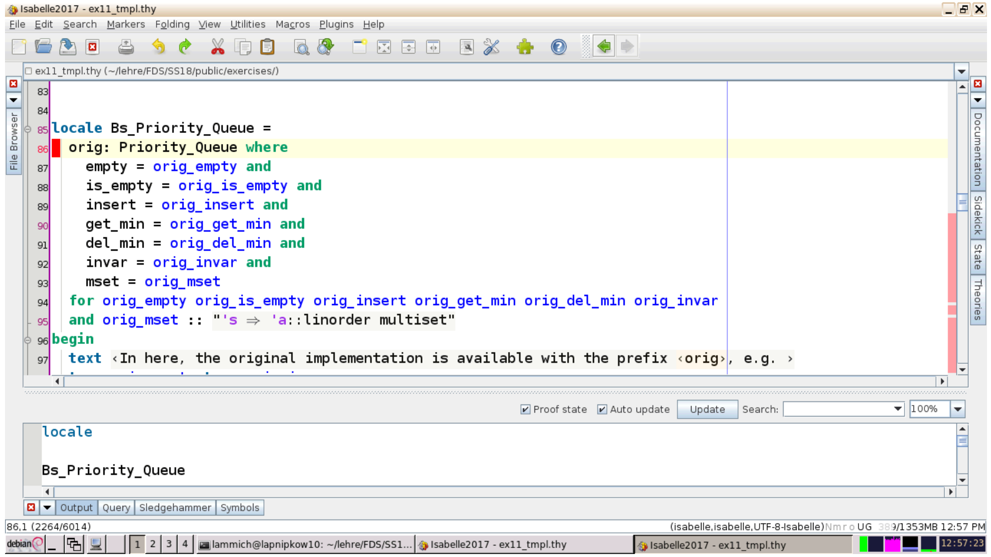

Isabelle2017 - ex11_tmpl.thy
ex11_tmpl.thy (~/lehre/FDS/SS18/public/exercises/)
text ‹
\Exercise{Bootstrapping a Priority Queue}
Given a generic priority queue implementation with
‹O(1)› ‹empty›, ‹is_empty› operations, ‹O($f_1$ n)› insert,
and ‹O($f_2$ n)› ‹get_min› and ‹del_min› operations.
Derive an implementation with ‹O(1)› ‹get_min›, and the
asymptotic complexities of the other operations unchanged!
Hint: Store the current minimal element! As you know nothing
about ‹$f_1$› and ‹$f_2$›, you must not use ‹get_min/del_min›
in your new ‹insert› operation, and vice versa!
Output Query Sledgehammer Symbols
69,1 (1732/6015)

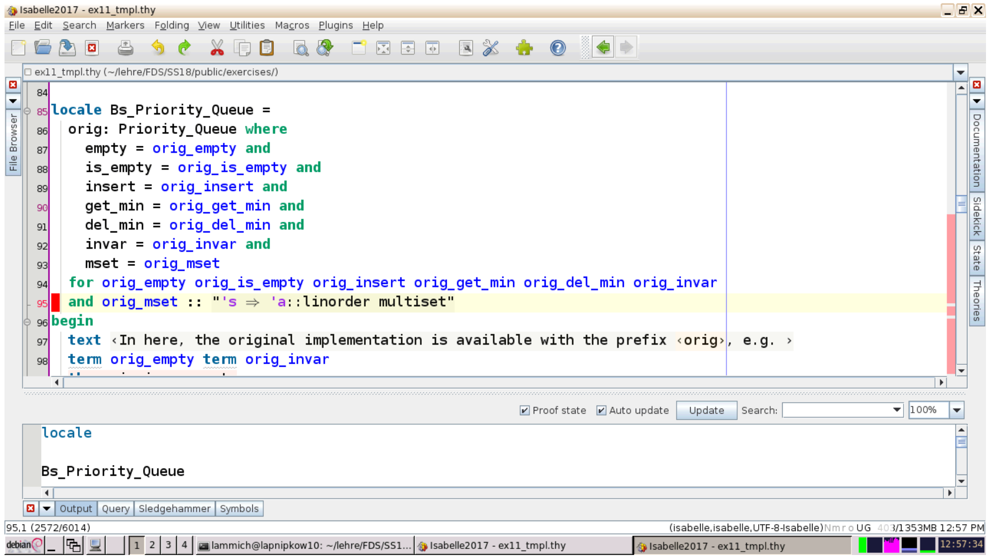

Isabelle2017 - ex11_tmpl.thy
ex11_tmpl.thy (~/lehre/FDS/SS18/public/exercises/)
text ‹
\Exercise{Bootstrapping a Priority Queue}
Given a generic priority queue implementation with
‹O(1)› ‹empty›, ‹is_empty› operations, ‹O($f_1$ n)› insert,
and ‹O($f_2$ n)› ‹get_min› and ‹del_min› operations.
Derive an implementation with ‹O(1)› ‹get_min›, and the
asymptotic complexities of the other operations unchanged!
Hint: Store the current minimal element! As you know nothing
about ‹$f_1$› and ‹$f_2$›, you must not use ‹get_min/del_min›
in your new ‹insert› operation, and vice versa!
Output Query Sledgehammer Symbols
69,18 (1749/6015)

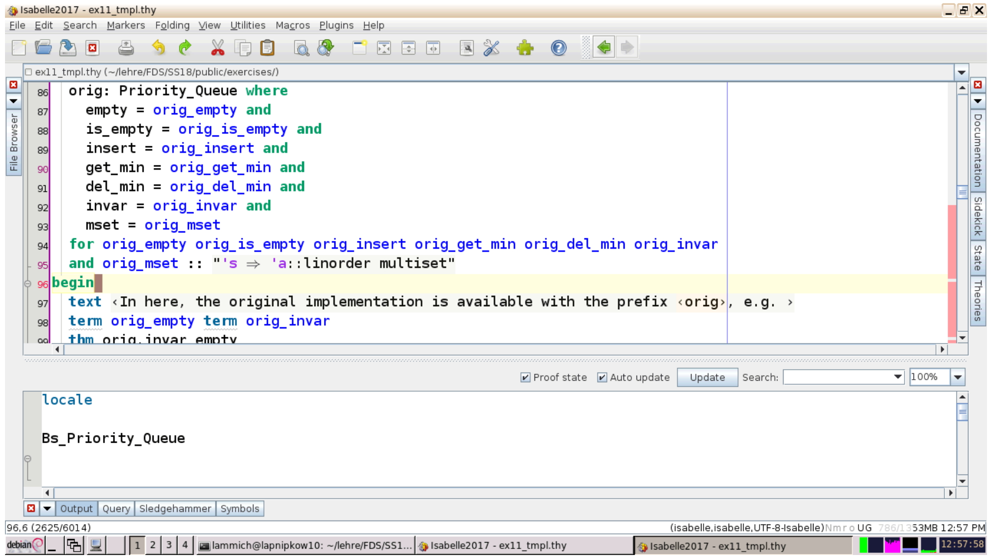

Isabelle2017 - ex11_tmpl.thy
ex11_tmpl.thy (~/lehre/FDS/SS18/public/exercises/)
Given a generic priority queue implementation with
‹O(1)› ‹empty›, ‹is_empty› operations, ‹O($f_1$ n)› insert,
and ‹O($f_2$ n)› ‹get_min› and ‹del_min› operations.
Derive an implementation with ‹O(1)› ‹get_min›, and the
asymptotic complexities of the other operations unchanged!
Hint: Store the current minimal element! As you know nothing
about ‹$f_1$› and ‹$f_2$›, you must not use ‹get_min/del_min›
in your new ‹insert› operation, and vice versa!
›
text ‹For technical reasons, you have to define the new implementations type
outside the locale!›
Output Query Sledgehammer Symbols
77,1 (2083/6015)

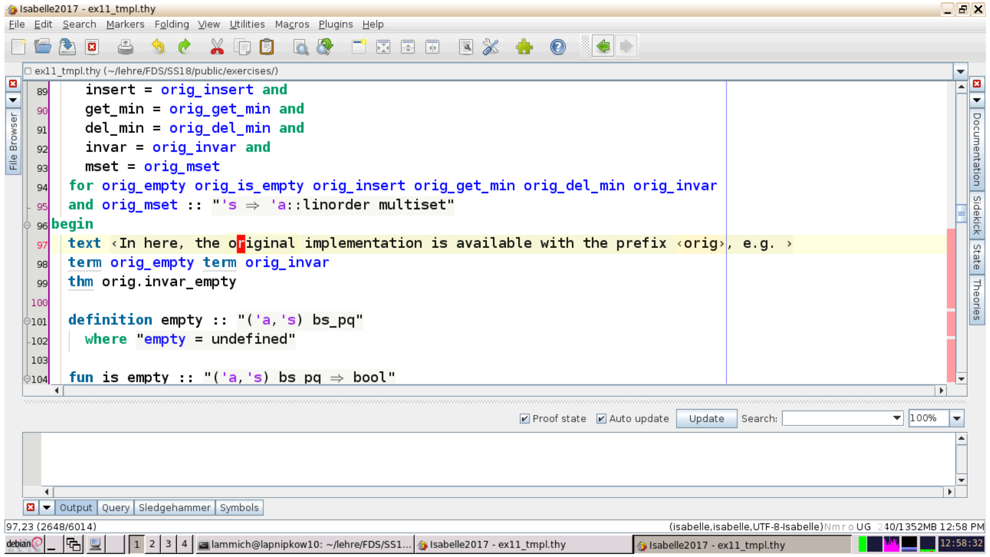

Isabelle2017 - ex11_tmpl.thy
ex11_tmpl.thy (~/lehre/FDS/SS18/public/exercises/)
Given a generic priority queue implementation with
‹O(1)› ‹empty›, ‹is_empty› operations, ‹O($f_1$ n)› insert,
and ‹O($f_2$ n)› ‹get_min› and ‹del_min› operations.
Derive an implementation with ‹O(1)› ‹get_min›, and the
asymptotic complexities of the other operations unchanged!
Hint: Store the current minimal element! As you know nothing
about ‹$f_1$› and ‹$f_2$›, you must not use ‹get_min/del_min›
in your new ‹insert› operation, and vice versa!
›
text ‹For technical reasons, you have to define the new implementations type
outside the locale!›
datatype ('a,'s) bs_pq = PUT_CONSTRUCTORS_HERE
Output Query Sledgehammer Symbols
78,1 (2085/6015)

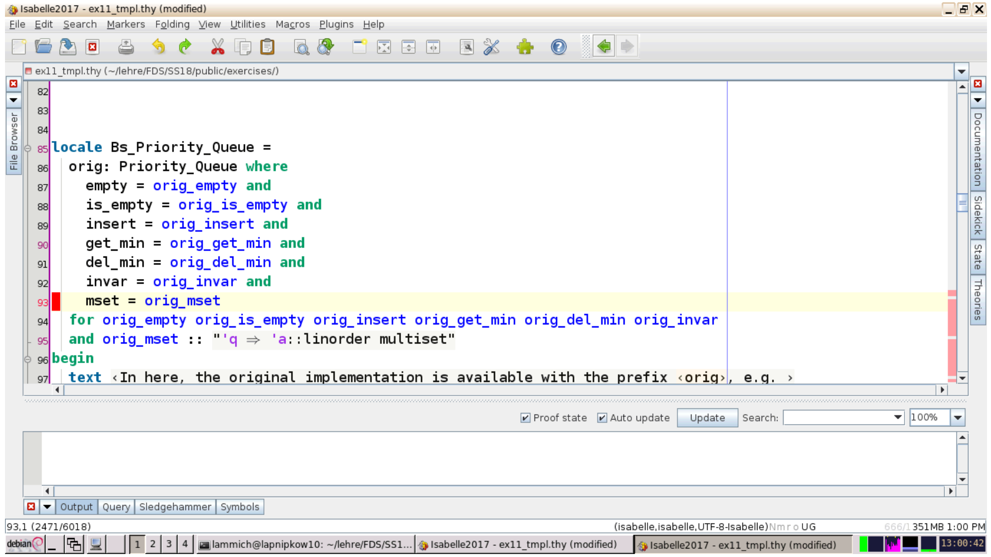

Isabelle2017 - ex11_tmpl.thy
asymptotic complexities of the other operations unchanged!
Hint: Store the current minimal element! As you know nothing
about ‹f₁› and ‹f₂›, you must not use ‹get_min/del_min›
in your new ‹insert› operation, and vice versa!
›
text ‹For technical reasons, you have to define the new implementations type
outside the locale!›
datatype ('a,'s) bs_pq = PUT_CONSTRUCTORS_HERE
locale Bs_Priority_Queue =
orig: Priority_Queue where
empty = orig_empty and
is_empty = orig_is_empty and
locale
Bs_Priority_Queue

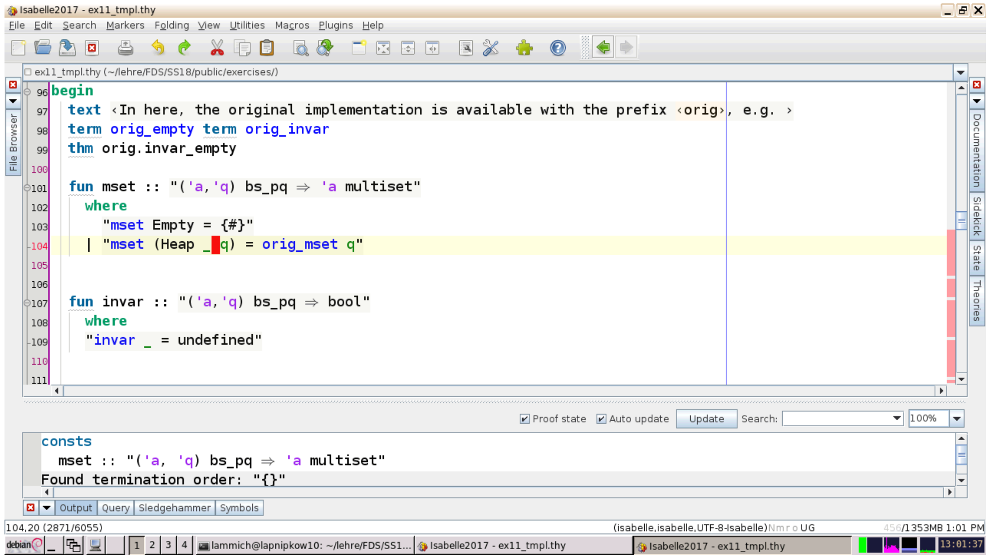

Isabelle2017 - ex11_tmpl.thy
locale Bs_Priority_Queue =
orig: Priority_Queue where
empty = orig_empty and
is_empty = orig_is_empty and
insert = orig_insert and
get_min = orig_get_min and
del_min = orig_del_min and
invar = orig_invar and
mset = orig_mset
for orig_empty orig_is_empty orig_insert orig_get_min orig_del_min orig_invar
and orig_mset :: "'s ⇒ 'a::linorder multiset"
begin
text ‹In here, the original implementation is available with the prefix ‹orig›, e.g. ›
locale
Bs_Priority_Queue

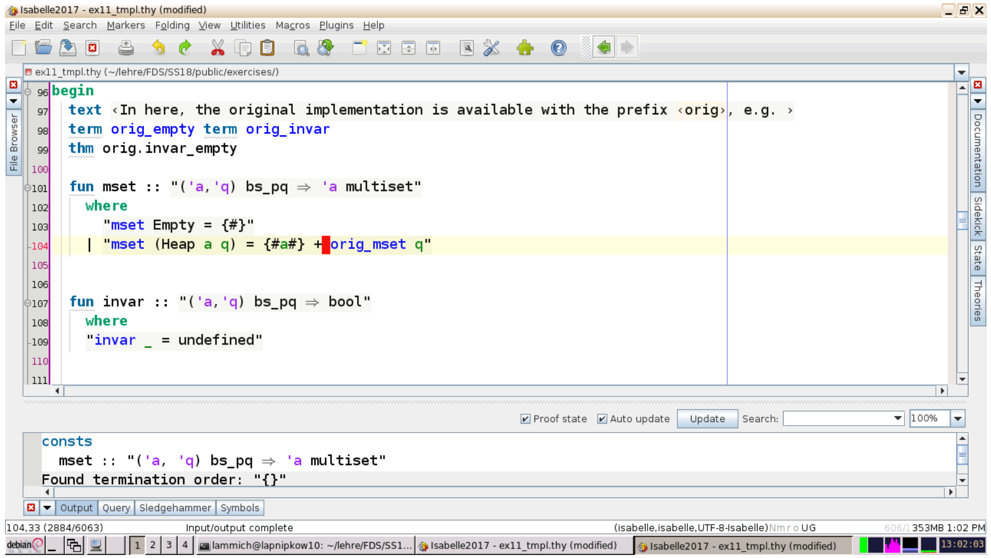

Isabelle2017 - ex11_tmpl.thy
locale Bs_Priority_Queue =
orig: Priority_Queue where
empty = orig_empty and
is_empty = orig_is_empty and
insert = orig_insert and
get_min = orig_get_min and
del_min = orig_del_min and
invar = orig_invar and
mset = orig_mset
for orig_empty orig_is_empty orig_insert orig_get_min orig_del_min orig_invar
and orig_mset :: "'s ⇒ 'a::linorder multiset"
begin
text ‹In here, the original implementation is available with the prefix ‹orig›, e.g. ›
term orig_empty term orig_invar
locale
Bs_Priority_Queue

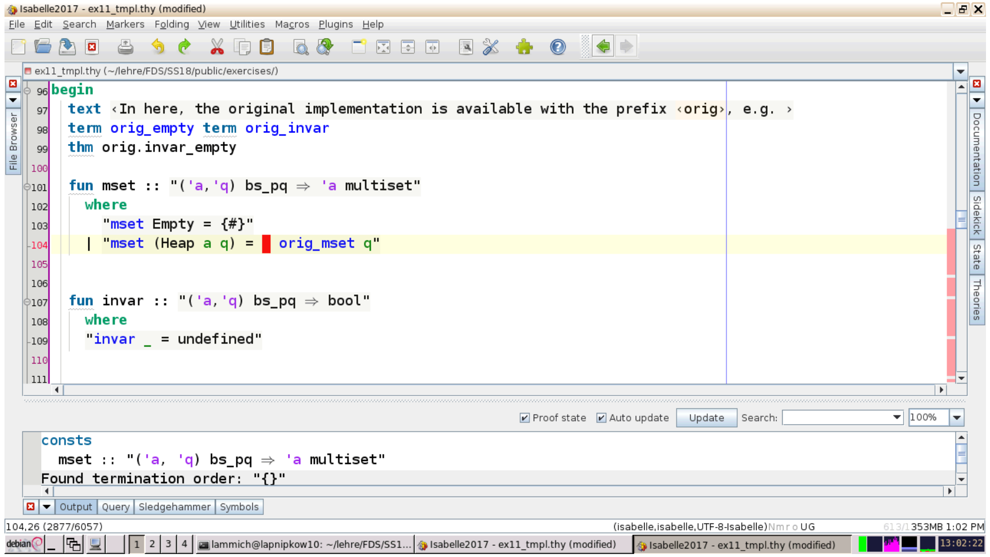
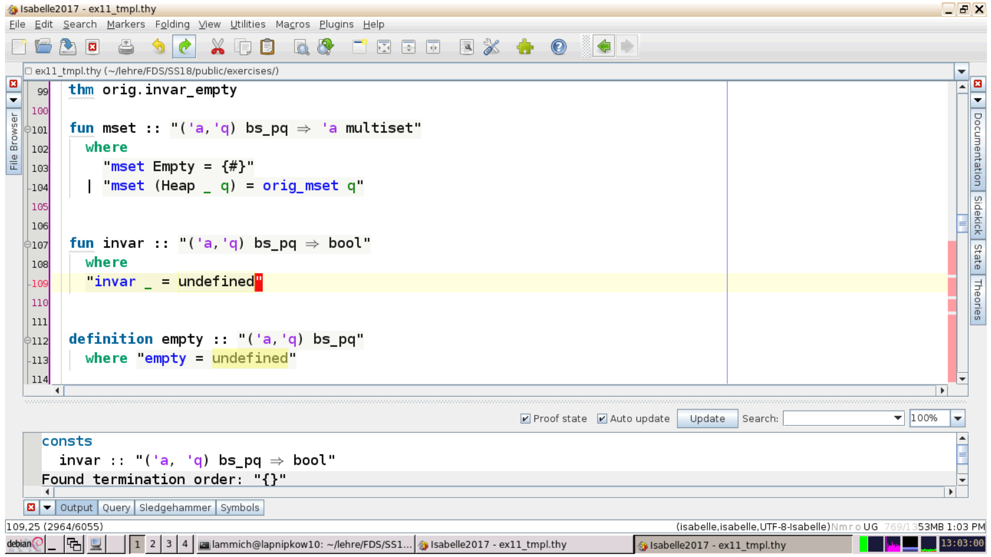
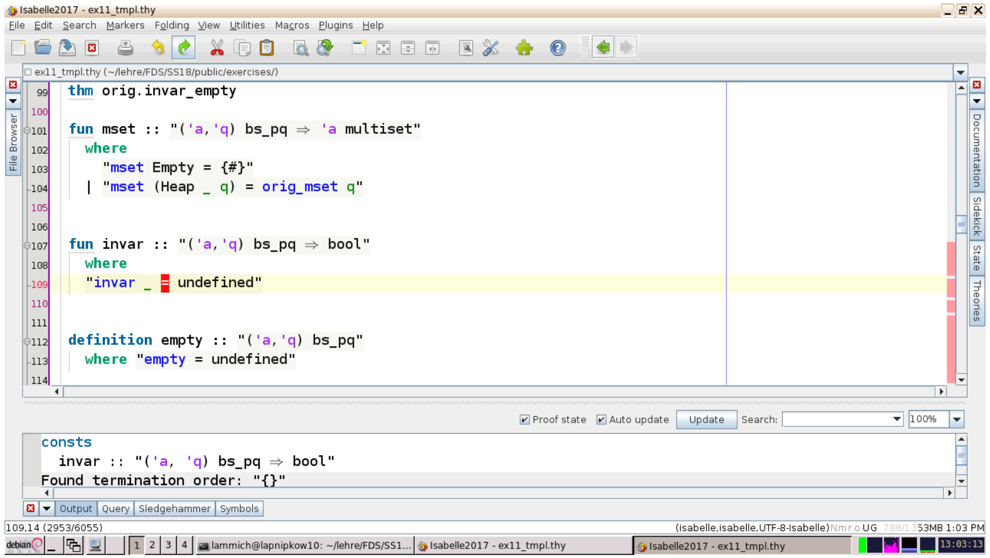
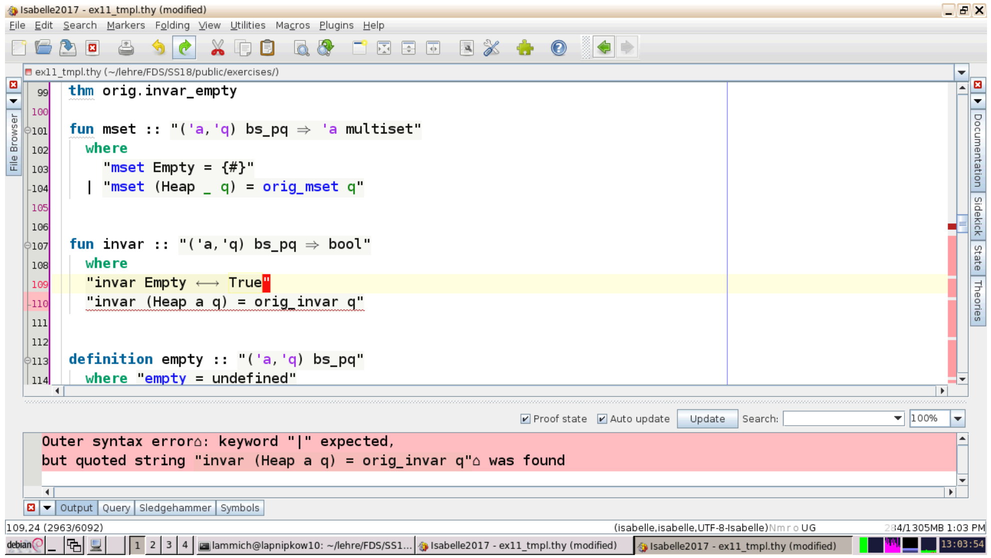

Isabelle2017 - ex11_tmpl.thy
orig: Priority_Queue where
empty = orig_empty and
is_empty = orig_is_empty and
insert = orig_insert and
get_min = orig_get_min and
del_min = orig_del_min and
invar = orig_invar and
mset = orig_mset
for orig_empty orig_is_empty orig_insert orig_get_min orig_del_min orig_invar
and orig_mset :: "'s ⇒ 'a::linorder multiset"
begin
text ‹In here, the original implementation is available with the prefix ‹orig›, e.g. ›
term orig_empty term orig_invar
thm orig.invar_empty
locale
Bs_Priority_Queue

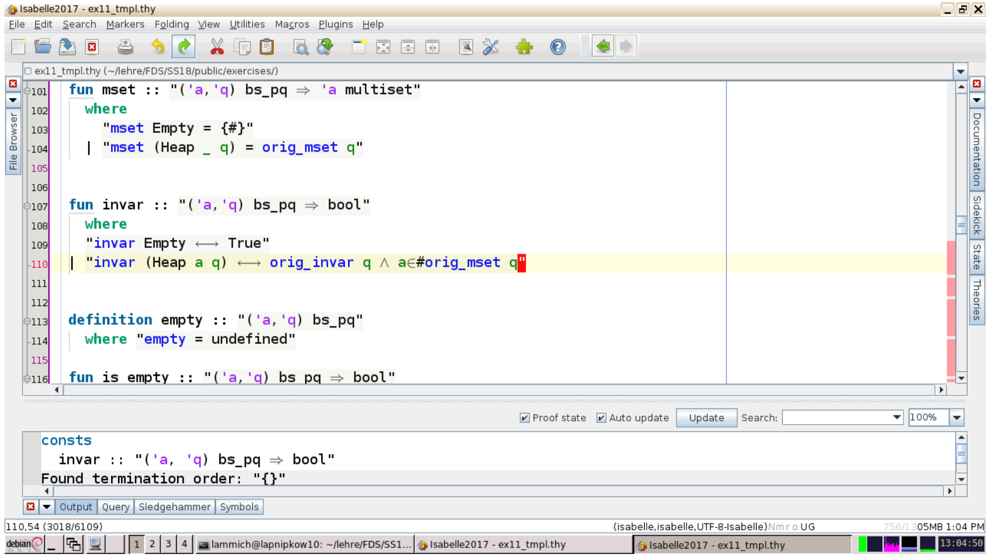
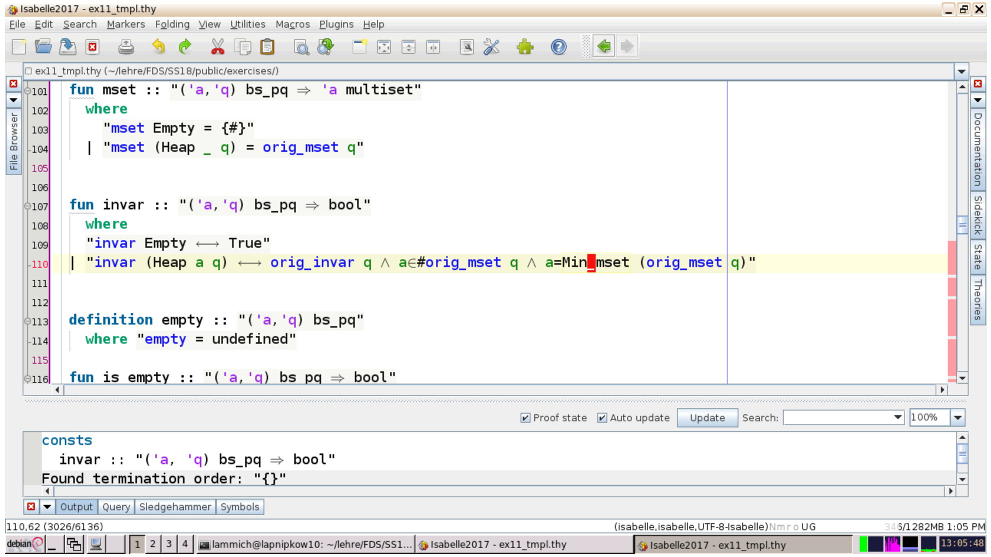
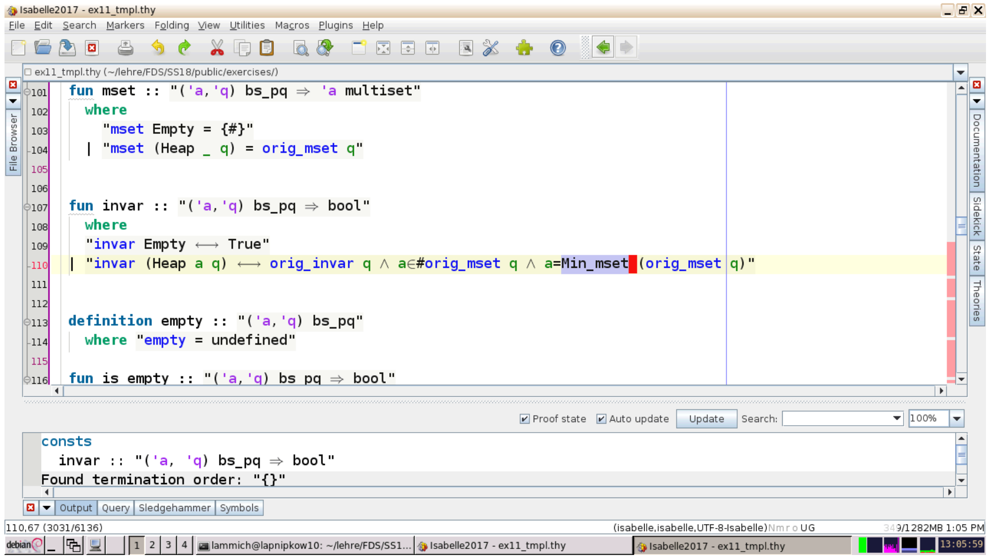
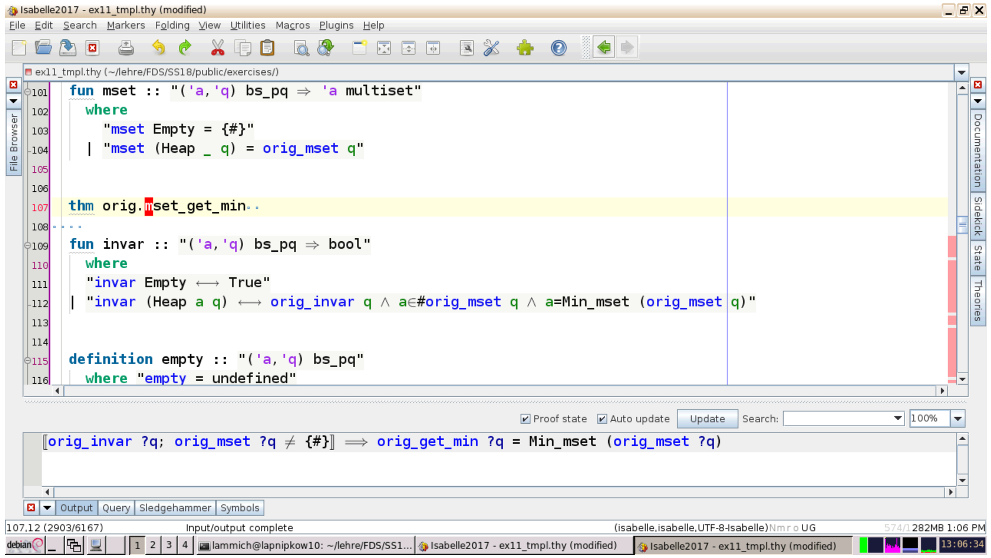

Isabelle2017 - ex11_tmpl.thy
insert = orig_insert and
get_min = orig_get_min and
del_min = orig_del_min and
invar = orig_invar and
mset = orig_mset
for orig_empty orig_is_empty orig_insert orig_get_min orig_del_min orig_invar
and orig_mset :: "'s ⇒ 'a::linorder multiset"
begin
text ‹In here, the original implementation is available with the prefix ‹orig›, e.g. ›
term orig_empty term orig_invar
thm orig.invar_empty
definition empty :: "('a,'s) bs_pq"
where "empty = undefined"
fun is_empty :: "('a,'s) bs_pq ⇒ bool"

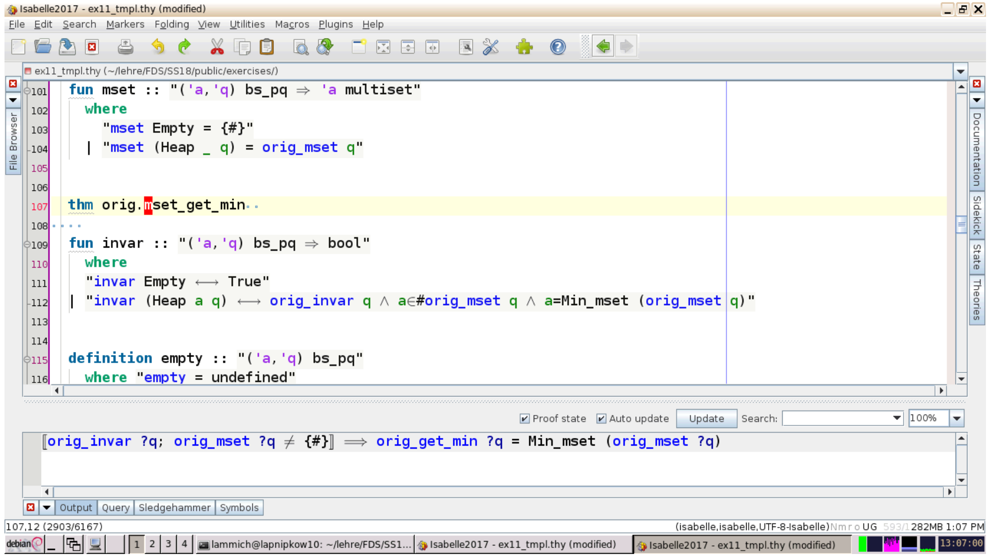

Isabelle2017 - ex11_tmpl.thy (modified)
locale Bs_Priority_Queue =
orig: Priority_Queue where
empty = orig_empty and
is_empty = orig_is_empty and
insert = orig_insert and
get_min = orig_get_min and
del_min = orig_del_min and
invar = orig_invar and
mset = orig_mset
for orig_empty orig_is_empty orig_insert orig_get_min orig_del_min orig_invar
and orig_mset :: "'q ⇒ 'a::linorder multiset"
begin
text ‹In here, the original implementation is available with the prefix ‹orig›, e.g. ›

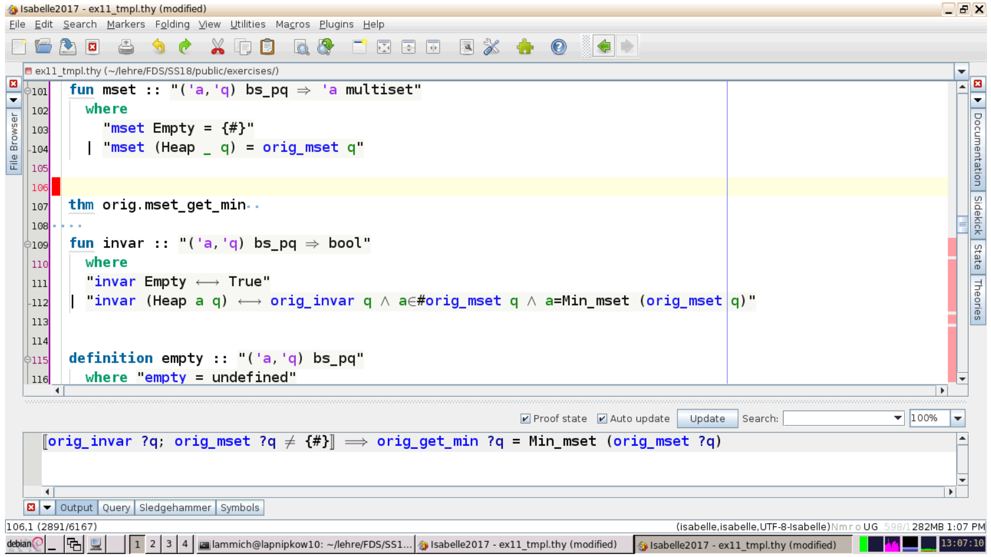

Isabelle2017 - ex11_tmpl.thy
begin
text ‹In here, the original implementation is available with the prefix ‹orig›, e.g. ›
term orig_empty term orig_invar
thm orig.invar_empty
fun mset :: "('a,'q) bs_pq ⇒ 'a multiset"
where
"mset Empty = {#}"
| "mset (Heap _ q) = orig_mset q"
fun invar :: "('a,'q) bs_pq ⇒ bool"
where
"invar _ = undefined"
consts
mset :: "('a, 'q) bs_pq ⇒ 'a multiset"
Found termination order: "{}"

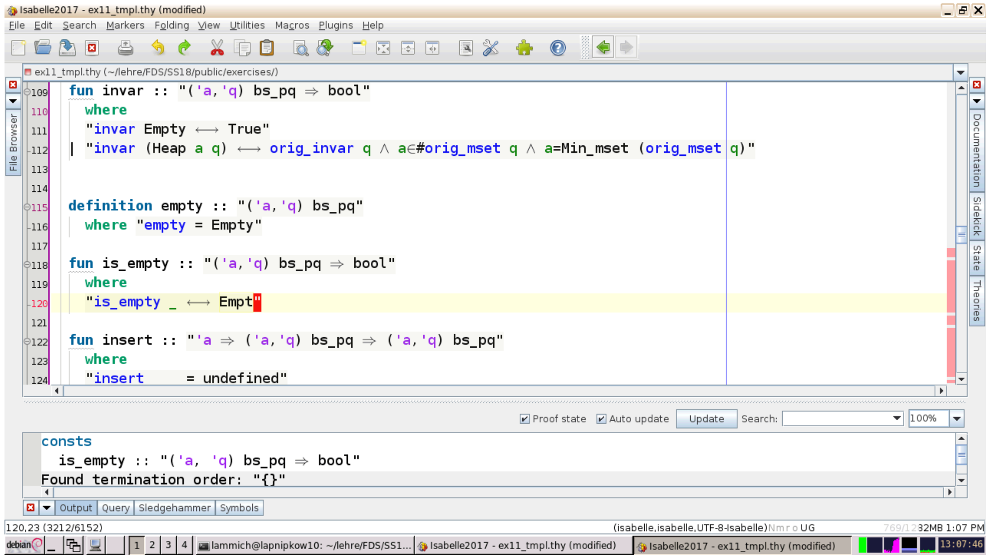

Isabelle2017 - ex11_tmpl.thy (modified)
begin
text ‹In here, the original implementation is available with the prefix ‹orig›, e.g. ›
term orig_empty term orig_invar
thm orig.invar_empty
fun mset :: "('a,'q) bs_pq ⇒ 'a multiset"
where
"mset Empty = {#}"
| "mset (Heap a q) = {#a#} + orig_mset q"
fun invar :: "('a,'q) bs_pq ⇒ bool"
where
"invar _ = undefined"
consts
mset :: "('a, 'q) bs_pq ⇒ 'a multiset"
Found termination order: "{}"

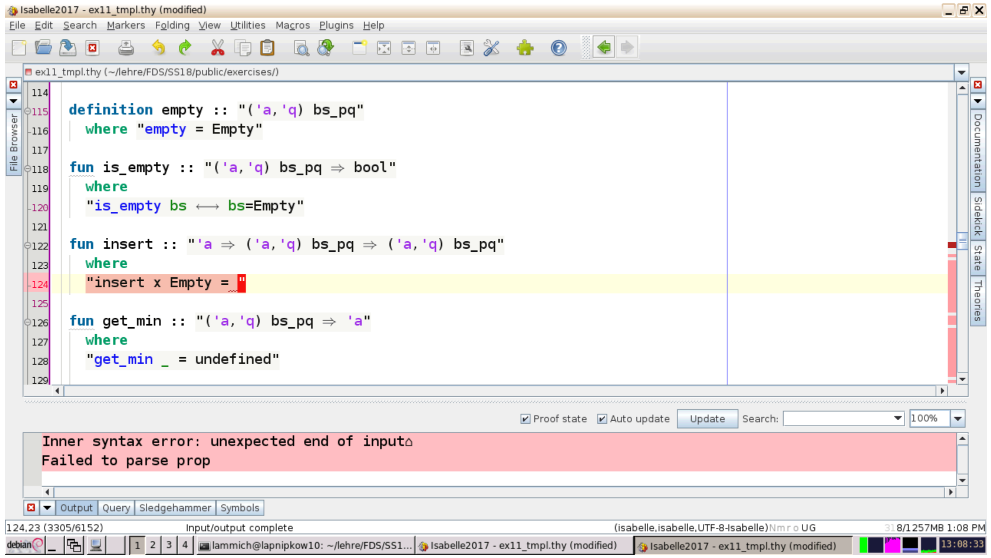

Isabelle2017 - ex11_tmpl.thy (modified)
File Edit Search Markers Folding View Utilities Macros Plugins Help
ex11_tmpl.thy (~/lehre/FDS/SS18/public/exercises/)
begin
text ‹In here, the original implementation is available with the prefix ‹orig›, e.g. ›
term orig_empty term orig_invar
thm orig.invar_empty
fun mset :: "('a,'q) bs_pq ⇒ 'a multiset"
where
"mset Empty = {#}"
| "mset (Heap a q) = orig_mset q"
fun invar :: "('a,'q) bs_pq ⇒ bool"
where
"invar _ = undefined"
Proof state
Auto update
Update
Search:
100%
consts
mset :: "('a, 'q) bs_pq ⇒ 'a multiset"
Found termination order: "{}"
Output
Query
Sledgehammer
Symbols
104,26 (2877/6057)
(isabelle,isabelle,UTF-8-Isabelle)
353MB 1:02 PM
lammich@lapnipkow10: ~/lehre/FDS/SS1...
Isabelle2017 - ex11_tmpl.thy (modified)
13:02:22

Isabelle2017 - ex11_tmpl.thy
File Edit Search Markers Folding View Utilities Macros Plugins Help
ex11_tmpl.thy (~/lehre/FDS/SS18/public/exercises/)
thm orig.invar_empty
fun mset :: "('a,'q) bs_pq ⇒ 'a multiset"
where
"mset Empty = {#}"
| "mset (Heap _ q) = orig_mset q"
fun invar :: "('a,'q) bs_pq ⇒ bool"
where
"invar _ = undefined"
definition empty :: "('a,'q) bs_pq"
where "empty = undefined"
Proof state
Auto update
Update
Search:
100%
consts
invar :: "('a, 'q) bs_pq ⇒ bool"
Found termination order: "{}"
Output
Query
Sledgehammer
Symbols
109,25 (2964/6055)
(isabelle,isabelle,UTF-8-Isabelle)
53MB 1:03 PM
lammich@lapnipkow10: ~/lehre/FDS/SS1...
Isabelle2017 - ex11_tmpl.thy
13:03:00

Isabelle2017 - ex11_tmpl.thy
File Edit Search Markers Folding View Utilities Macros Plugins Help
ex11_tmpl.thy (~/lehre/FDS/SS18/public/exercises/)
thm orig.invar_empty
fun mset :: "('a,'q) bs_pq ⇒ 'a multiset"
where
"mset Empty = {#}"
| "mset (Heap _ q) = orig_mset q"
fun invar :: "('a,'q) bs_pq ⇒ bool"
where
"invar _ = undefined"
definition empty :: "('a,'q) bs_pq"
where "empty = undefined"
Proof state
Auto update
Update
Search:
100%
consts
invar :: "('a, 'q) bs_pq ⇒ bool"
Found termination order: "{}"
Output
Query
Sledgehammer
Symbols
109,14 (2953/6055)
(isabelle,isabelle,UTF-8-Isabelle)
53MB 1:03 PM
lammich@lapnipkow10: ~/lehre/FDS/SS1...
Isabelle2017 - ex11_tmpl.thy
13:03:13

Isabelle2017 - ex11_tmpl.thy (modified)
File Edit Search Markers Folding View Utilities Macros Plugins Help
ex11_tmpl.thy (~/lehre/FDS/SS18/public/exercises/)
thm orig.invar_empty
fun mset :: "('a,'q) bs_pq ⇒ 'a multiset"
where
"mset Empty = {#}"
| "mset (Heap _ q) = orig_mset q"
fun invar :: "('a,'q) bs_pq ⇒ bool"
where
"invar Empty ⟷ True"
"invar (Heap a q) = orig_invar q"
definition empty :: "('a,'q) bs_pq"
where "empty = undefined"
Proof state
Auto update
Update
Search:
100%
Outer syntax error⌂: keyword "|" expected,
but quoted string "invar (Heap a q) = orig_invar q"⌂ was found
Output
Query
Sledgehammer
Symbols
109,24 (2963/6092)
(isabelle,isabelle,UTF-8-Isabelle)
1:03 PM
lammich@lapnipkow10: ~/lehre/FDS/SS1...
Isabelle2017 - ex11_tmpl.thy (modified)
13:03:54

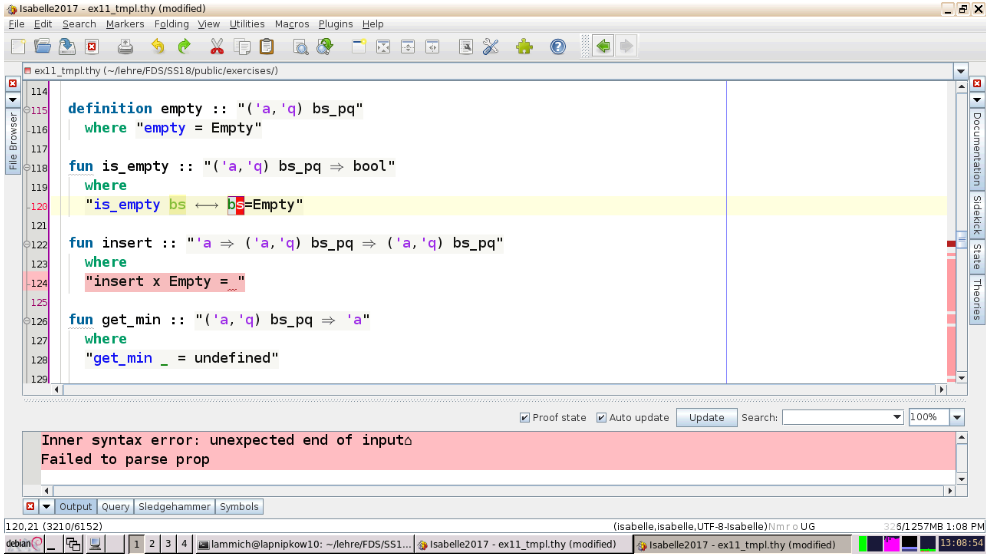

Isabelle2017 - ex11_tmpl.thy
ex11_tmpl.thy (~/lehre/FDS/SS18/public/exercises/)
fun mset :: "('a,'q) bs_pq ⇒ 'a multiset"
where
"mset Empty = {#}"
| "mset (Heap _ q) = orig_mset q"
fun invar :: "('a,'q) bs_pq ⇒ bool"
where
"invar Empty ⟷ True"
| "invar (Heap a q) ⟷ orig_invar q ∧ a∈#orig_mset q"
definition empty :: "('a,'q) bs_pq"
where "empty = undefined"
fun is_empty :: "('a,'q) bs_pq ⇒ bool"
consts
invar :: "('a, 'q) bs_pq ⇒ bool"
Found termination order: "{}"

Isabelle2017 - ex11_tmpl.thy
ex11_tmpl.thy (~/lehre/FDS/SS18/public/exercises/)
fun mset :: "('a,'q) bs_pq ⇒ 'a multiset"
where
"mset Empty = {#}"
| "mset (Heap _ q) = orig_mset q"
fun invar :: "('a,'q) bs_pq ⇒ bool"
where
"invar Empty ⟷ True"
| "invar (Heap a q) ⟷ orig_invar q ∧ a∈#orig_mset q ∧ a=Min_mset (orig_mset q)"
definition empty :: "('a,'q) bs_pq"
where "empty = undefined"
fun is_empty :: "('a,'q) bs_pq ⇒ bool"
consts
invar :: "('a, 'q) bs_pq ⇒ bool"
Found termination order: "{}"

Isabelle2017 - ex11_tmpl.thy
ex11_tmpl.thy (~/lehre/FDS/SS18/public/exercises/)
fun mset :: "('a,'q) bs_pq ⇒ 'a multiset"
where
"mset Empty = {#}"
| "mset (Heap _ q) = orig_mset q"
fun invar :: "('a,'q) bs_pq ⇒ bool"
where
"invar Empty ⟷ True"
| "invar (Heap a q) ⟷ orig_invar q ∧ a∈#orig_mset q ∧ a=Min_mset (orig_mset q)"
definition empty :: "('a,'q) bs_pq"
where "empty = undefined"
fun is_empty :: "('a,'q) bs_pq ⇒ bool"
consts
invar :: "('a, 'q) bs_pq ⇒ bool"
Found termination order: "{}"

Isabelle2017 - ex11_tmpl.thy (modified)
ex11_tmpl.thy (~/lehre/FDS/SS18/public/exercises/)
fun mset :: "('a,'q) bs_pq ⇒ 'a multiset"
where
"mset Empty = {#}"
| "mset (Heap _ q) = orig_mset q"
thm orig.mset_get_min
fun invar :: "('a,'q) bs_pq ⇒ bool"
where
"invar Empty ⟷ True"
| "invar (Heap a q) ⟷ orig_invar q ∧ a∈#orig_mset q ∧ a=Min_mset (orig_mset q)"
definition empty :: "('a,'q) bs_pq"
where "empty = undefined"
⟦orig_invar ?q; orig_mset ?q ≠ {#}⟧ ⟹ orig_get_min ?q = Min_mset (orig_mset ?q)

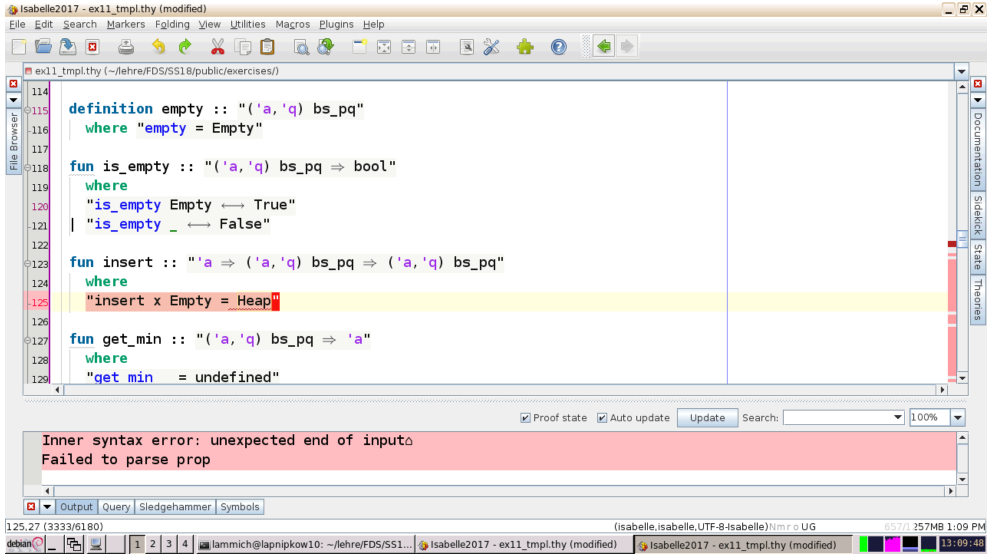

Isabelle2017 - ex11_tmpl.thy (modified)
fun mset :: "('a,'q) bs_pq ⇒ 'a multiset"
where
"mset Empty = {#}"
| "mset (Heap _ q) = orig_mset q"
thm orig.mset_get_min
fun invar :: "('a,'q) bs_pq ⇒ bool"
where
"invar Empty ⟷ True"
| "invar (Heap a q) ⟷ orig_invar q ∧ a∈#orig_mset q ∧ a=Min_mset (orig_mset q)"
definition empty :: "('a,'q) bs_pq"
where "empty = undefined"
⟦orig_invar ?q; orig_mset ?q ≠ {#}⟧ ⟹ orig_get_min ?q = Min_mset (orig_mset ?q)

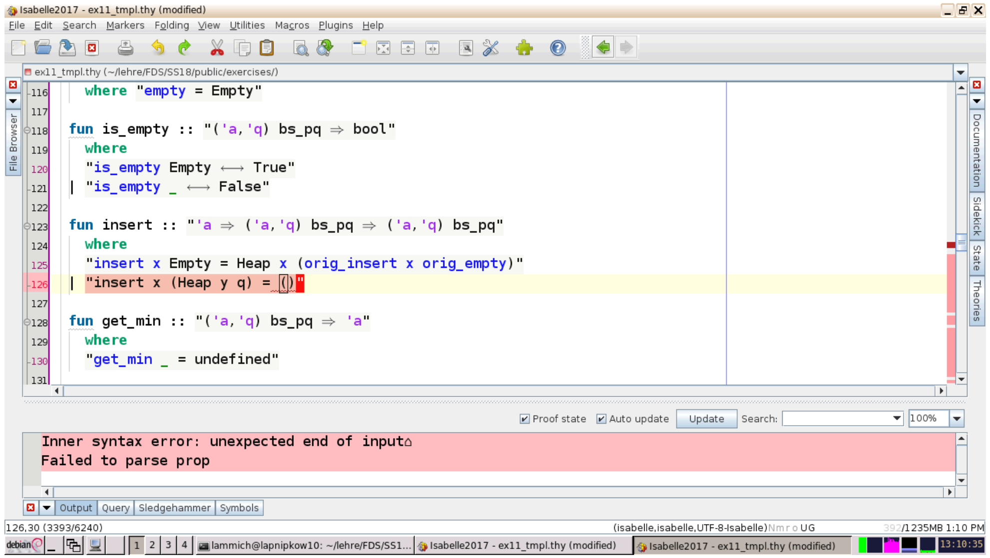

Isabelle2017 - ex11_tmpl.thy (modified)
fun mset :: "('a,'q) bs_pq ⇒ 'a multiset"
where
"mset Empty = {#}"
| "mset (Heap _ q) = orig_mset q"
thm orig.mset_get_min
fun invar :: "('a,'q) bs_pq ⇒ bool"
where
"invar Empty ⟷ True"
| "invar (Heap a q) ⟷ orig_invar q ∧ a∈#orig_mset q ∧ a=Min_mset (orig_mset q)"
definition empty :: "('a,'q) bs_pq"
where "empty = undefined"
⟦orig_invar ?q; orig_mset ?q ≠ {#}⟧ ⟹ orig_get_min ?q = Min_mset (orig_mset ?q)

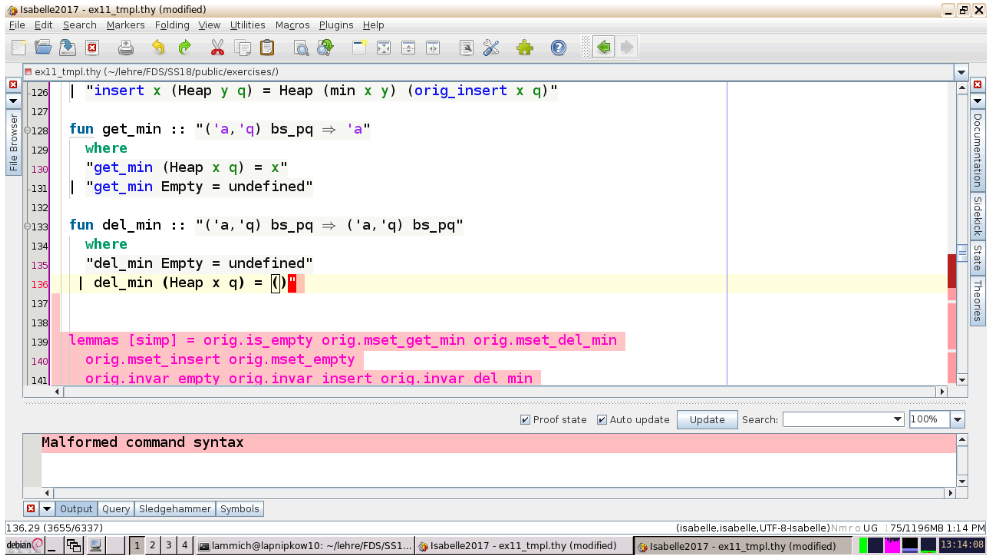

Isabelle2017 - ex11_tmpl.thy (modified)
fun invar :: "('a,'q) bs_pq ⇒ bool"
where
"invar Empty ⟷ True"
| "invar (Heap a q) ⟷ orig_invar q ∧ a∈#orig_mset q ∧ a=Min_mset (orig_mset q)"
definition empty :: "('a,'q) bs_pq"
where "empty = Empty"
fun is_empty :: "('a,'q) bs_pq ⇒ bool"
where
"is_empty _ ⟷ Empty"
fun insert :: "'a ⇒ ('a,'q) bs_pq ⇒ ('a,'q) bs_pq"
where
"insert = undefined"
consts
is_empty :: "('a, 'q) bs_pq ⇒ bool"
Found termination order: "{}"

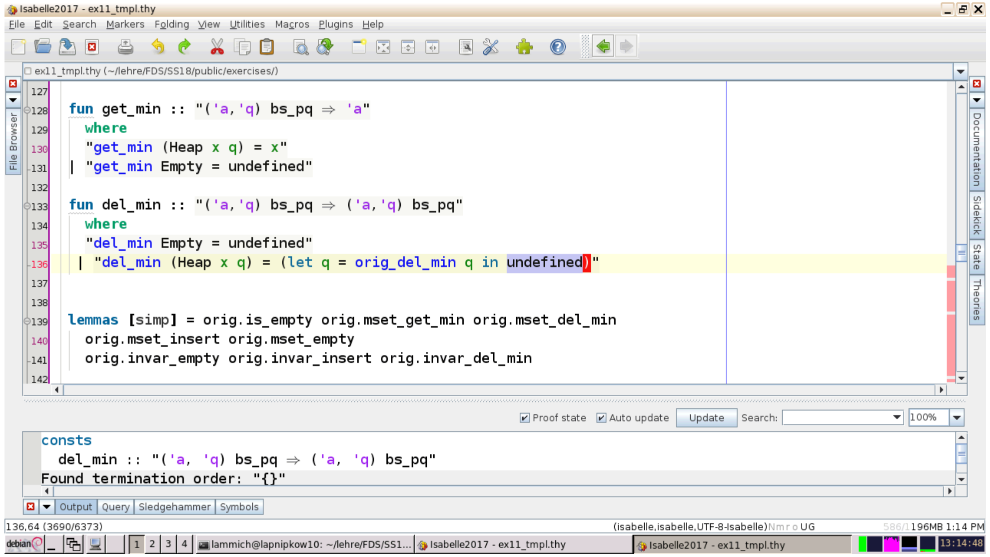

Isabelle2017 - ex11_tmpl.thy (modified)
definition empty :: "('a,'q) bs_pq"
where "empty = Empty"
fun is_empty :: "('a,'q) bs_pq ⇒ bool"
where
"is_empty bs ⟷ bs=Empty"
fun insert :: "'a ⇒ ('a,'q) bs_pq ⇒ ('a,'q) bs_pq"
where
"insert x Empty = "
fun get_min :: "('a,'q) bs_pq ⇒ 'a"
where
"get_min _ = undefined"
Inner syntax error: unexpected end of input
Failed to parse prop

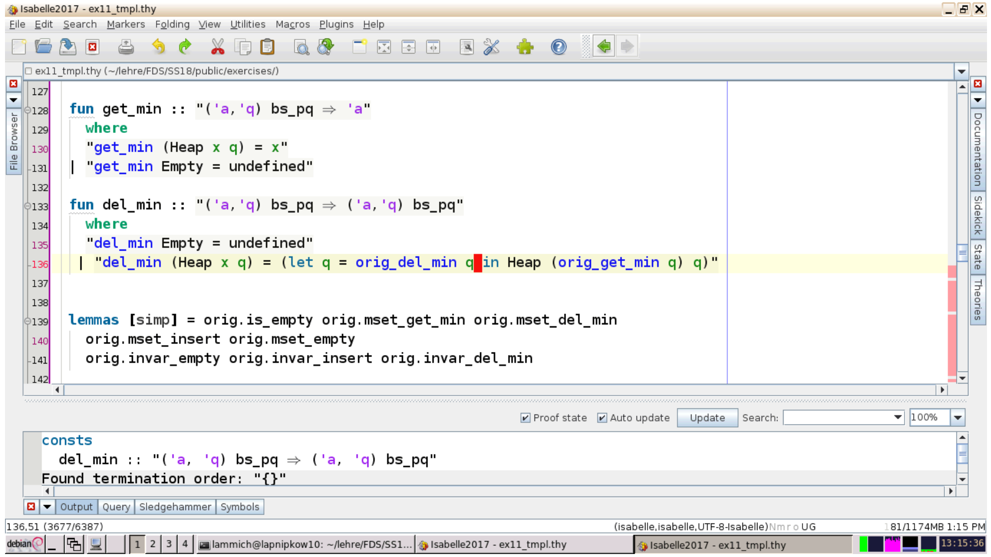

Isabelle2017 - ex11_tmpl.thy (modified)
definition empty :: "('a,'q) bs_pq"
where "empty = Empty"
fun is_empty :: "('a,'q) bs_pq ⇒ bool"
where
"is_empty bs ⟷ bs=Empty"
fun insert :: "'a ⇒ ('a,'q) bs_pq ⇒ ('a,'q) bs_pq"
where
"insert x Empty = _"
fun get_min :: "('a,'q) bs_pq ⇒ 'a"
where
"get_min _ = undefined"
Inner syntax error: unexpected end of input
Failed to parse prop

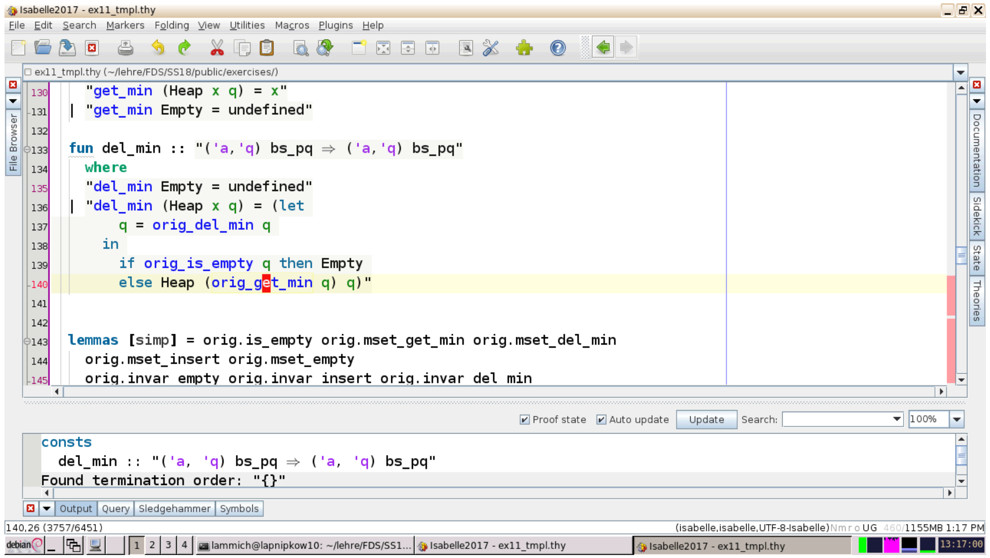

Isabelle2017 - ex11_tmpl.thy (modified)
definition empty :: "('a,'q) bs_pq"
where "empty = Empty"
fun is_empty :: "('a,'q) bs_pq ⇒ bool"
where
"is_empty Empty ⟷ True"
| "is_empty _ ⟷ False"
fun insert :: "'a ⇒ ('a,'q) bs_pq ⇒ ('a,'q) bs_pq"
where
"insert x Empty = Heap"
fun get_min :: "('a,'q) bs_pq ⇒ 'a"
where
"get_min _ = undefined"
Inner syntax error: unexpected end of input
Failed to parse prop

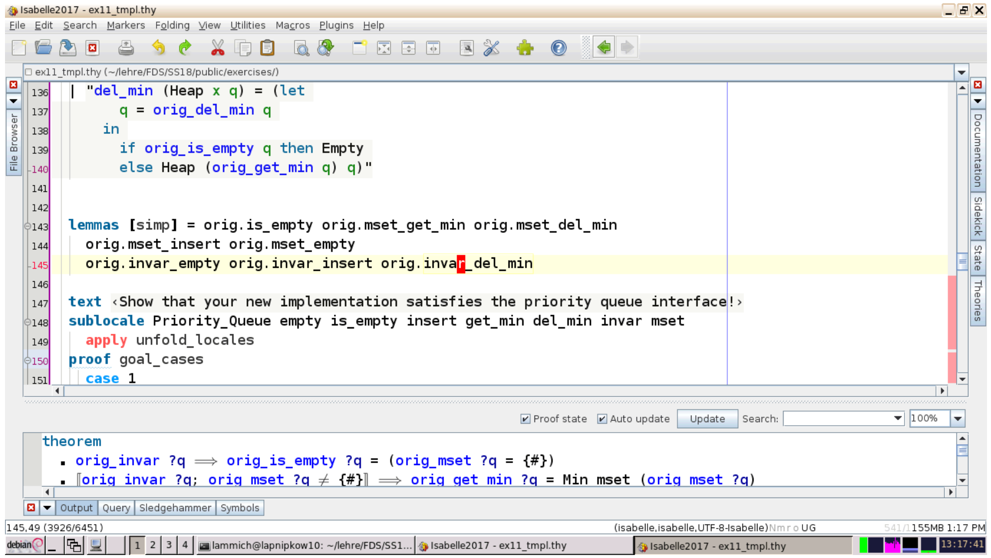

Isabelle2017 - ex11_tmpl.thy (modified)
where "empty = Empty"
fun is_empty :: "('a,'q) bs_pq ⇒ bool"
where
"is_empty Empty ⟷ True"
| "is_empty _ ⟷ False"
fun insert :: "'a ⇒ ('a,'q) bs_pq ⇒ ('a,'q) bs_pq"
where
"insert x Empty = Heap x (orig_insert x orig_empty)"
| "insert x (Heap y q) = ()"
fun get_min :: "('a,'q) bs_pq ⇒ 'a"
where
"get_min _ = undefined"
Inner syntax error: unexpected end of input
Failed to parse prop

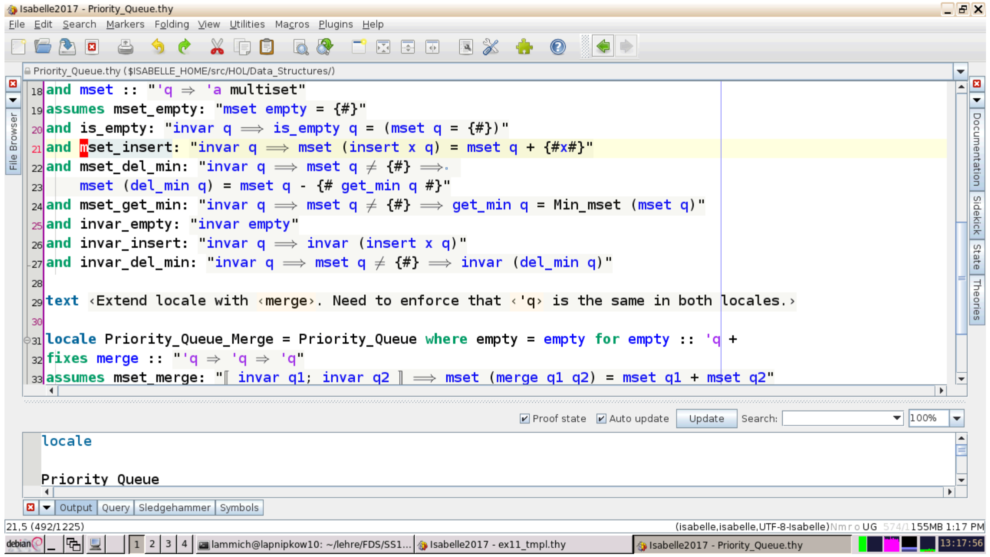
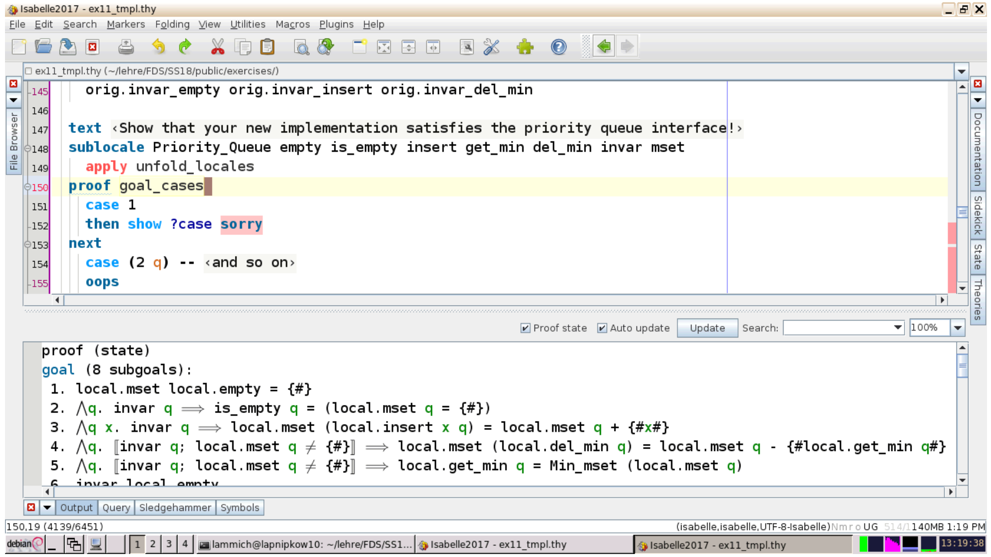
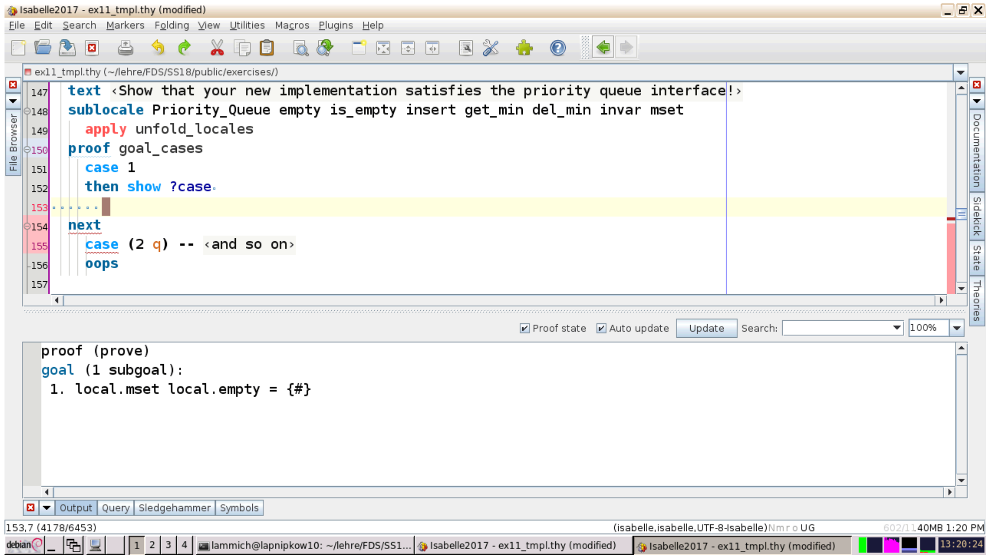
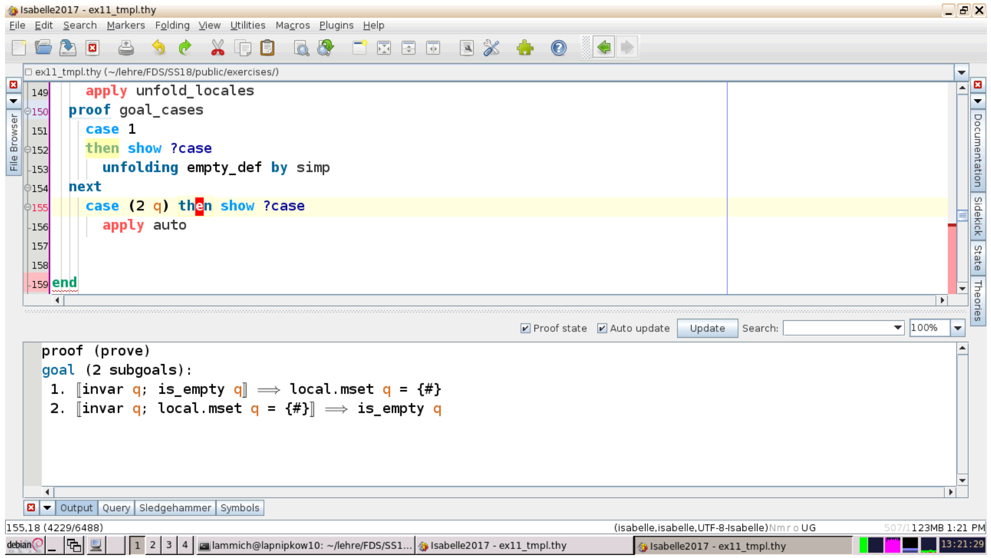

Isabelle2017 - ex11_tmpl.thy (modified)
| "insert x (Heap y q) = Heap (min x y) (orig_insert x q)"
fun get_min :: "('a,'q) bs_pq ⇒ 'a"
where
"get_min (Heap x q) = x"
| "get_min Empty = undefined"
fun del_min :: "('a,'q) bs_pq ⇒ ('a,'q) bs_pq"
where
"del_min Empty = undefined"
| del_min (Heap x q) = ()"
lemmas [simp] = orig.is_empty orig.mset_get_min orig.mset_del_min
orig.mset_insert orig.mset_empty
orig.invar_empty orig.invar_insert orig.invar_del_min
Malformed command syntax

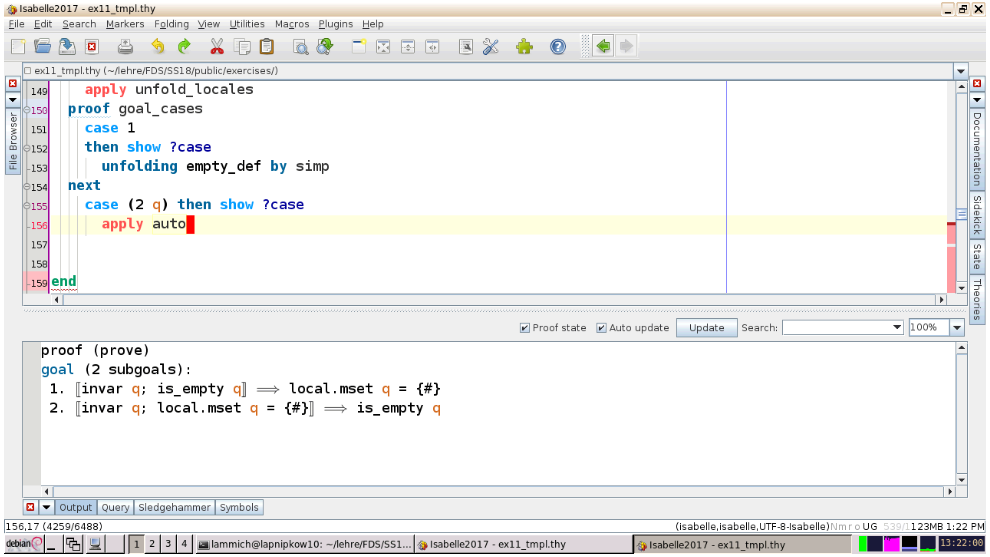

Isabelle2017 - ex11_tmpl.thy
fun get_min :: "('a,'q) bs_pq ⇒ 'a"
where
"get_min (Heap x q) = x"
| "get_min Empty = undefined"
fun del_min :: "('a,'q) bs_pq ⇒ ('a,'q) bs_pq"
where
"del_min Empty = undefined"
| "del_min (Heap x q) = (let q = orig_del_min q in undefined)"
lemmas [simp] = orig.is_empty orig.mset_get_min orig.mset_del_min
orig.mset_insert orig.mset_empty
orig.invar_empty orig.invar_insert orig.invar_del_min
consts
del_min :: "('a, 'q) bs_pq ⇒ ('a, 'q) bs_pq"
Found termination order: "{}"

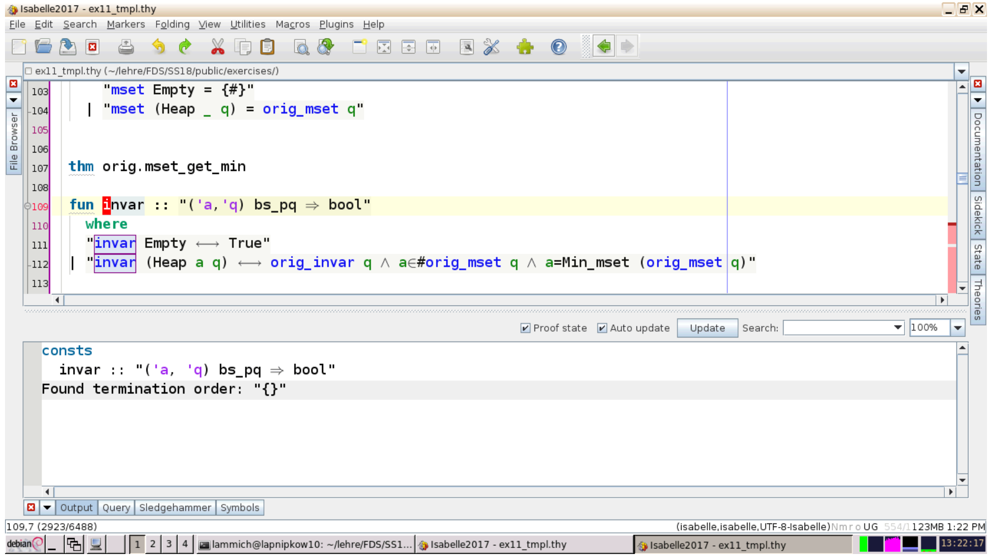

Isabelle2017 - ex11_tmpl.thy
fun get_min :: "('a,'q) bs_pq ⇒ 'a"
where
"get_min (Heap x q) = x"
| "get_min Empty = undefined"
fun del_min :: "('a,'q) bs_pq ⇒ ('a,'q) bs_pq"
where
"del_min Empty = undefined"
| "del_min (Heap x q) = (let q = orig_del_min q in Heap (orig_get_min q) q)"
lemmas [simp] = orig.is_empty orig.mset_get_min orig.mset_del_min
orig.mset_insert orig.mset_empty
orig.invar_empty orig.invar_insert orig.invar_del_min
consts
del_min :: "('a, 'q) bs_pq ⇒ ('a, 'q) bs_pq"
Found termination order: "{}"

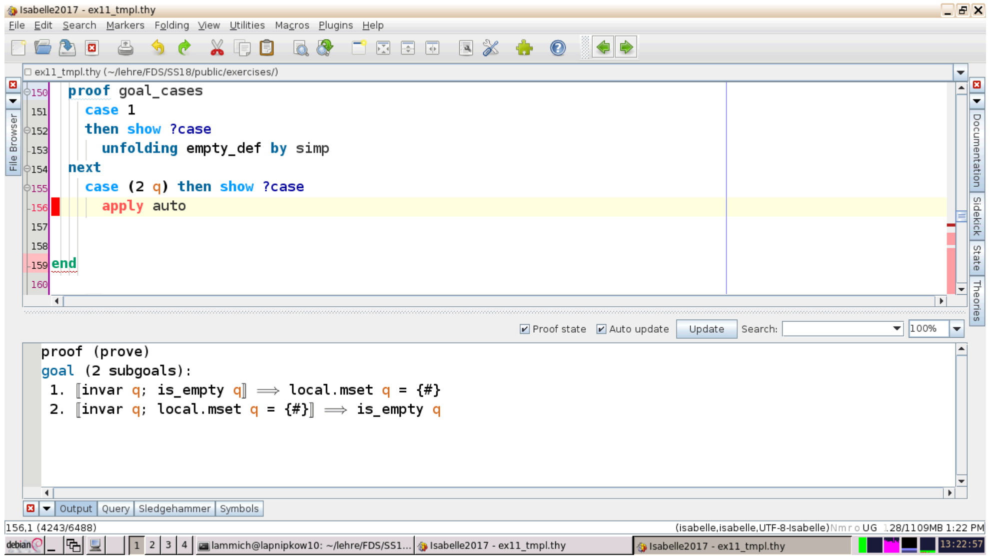

Isabelle2017 - ex11_tmpl.thy
"get_min (Heap x q) = x"
| "get_min Empty = undefined"
fun del_min :: "('a,'q) bs_pq ⇒ ('a,'q) bs_pq"
where
"del_min Empty = undefined"
| "del_min (Heap x q) = (let
q = orig_del_min q
in
if orig_is_empty q then Empty
else Heap (orig_get_min q) q)"
lemmas [simp] = orig.is_empty orig.mset_get_min orig.mset_del_min
orig.mset_insert orig.mset_empty
orig.invar_empty orig.invar_insert orig.invar_del_min
consts
del_min :: "('a, 'q) bs_pq ⇒ ('a, 'q) bs_pq"
Found termination order: "{}"

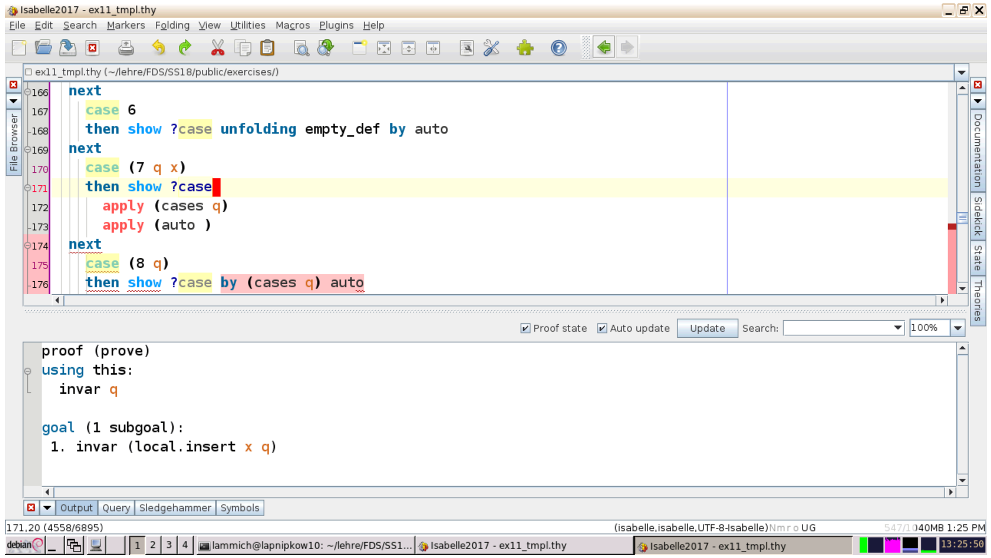

Isabelle2017 - ex11_tmpl.thy
| "del_min (Heap x q) = (let
q = orig_del_min q
in
if orig_is_empty q then Empty
else Heap (orig_get_min q) q)"
lemmas [simp] = orig.is_empty orig.mset_get_min orig.mset_del_min
orig.mset_insert orig.mset_empty
orig.invar_empty orig.invar_insert orig.invar_del_min
text ‹Show that your new implementation satisfies the priority queue interface!›
sublocale Priority_Queue empty is_empty insert get_min del_min invar mset
apply unfold_locales
proof goal_cases
case 1
theorem
• orig_invar ?q ⟹ orig_is_empty ?q = (orig_mset ?q = {#})
• ⟦orig_invar ?q; orig_mset ?q ≠ {#}⟧ ⟹ orig_get_min ?q = Min_mset (orig_mset ?q)

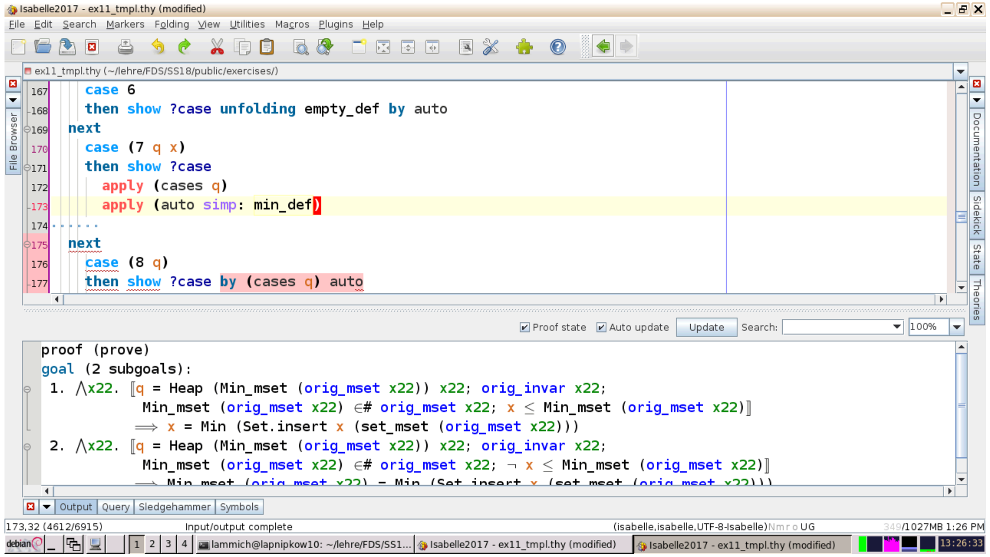
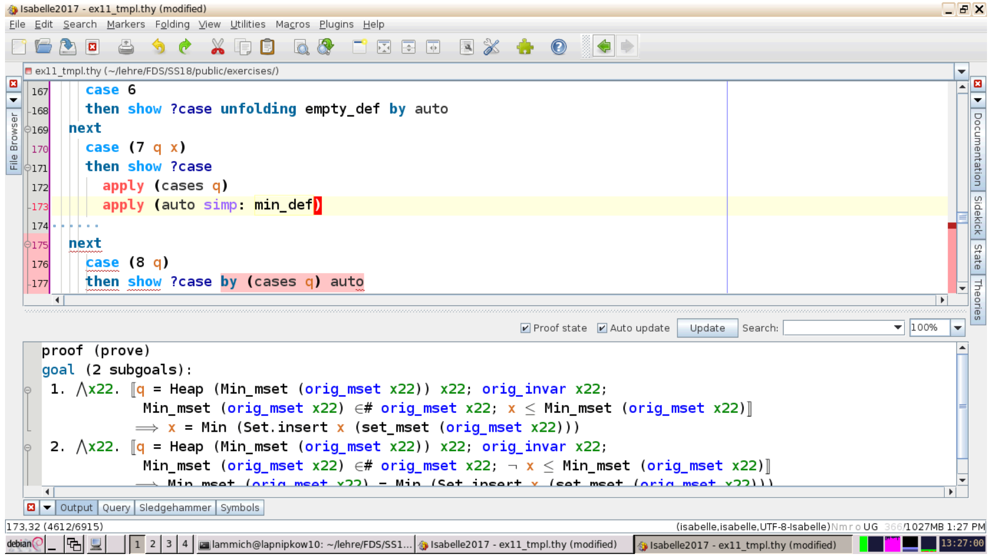
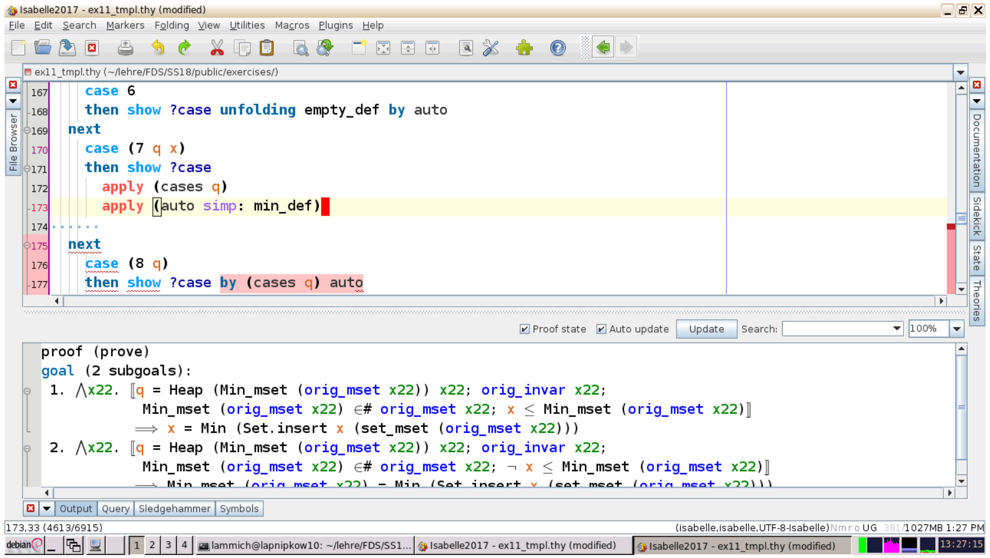
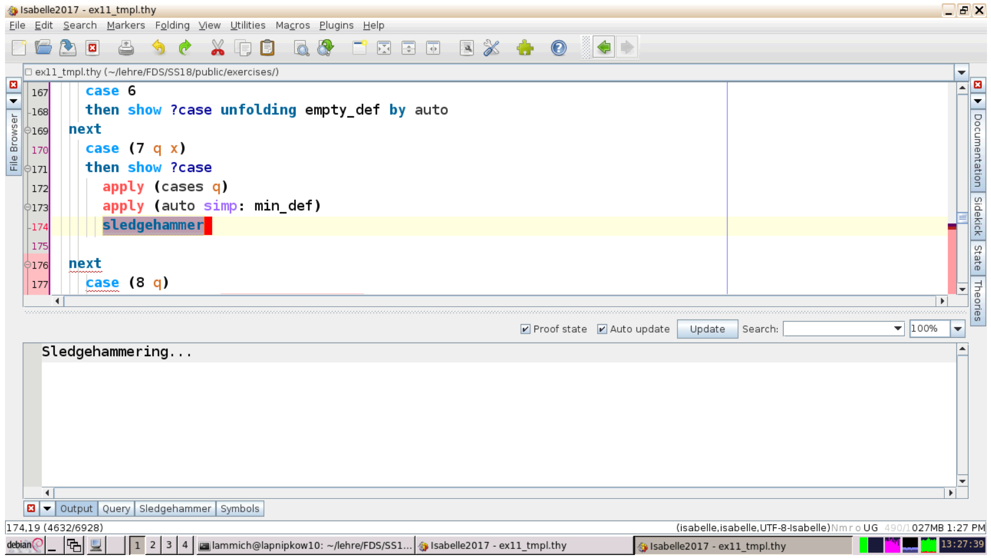

Isabelle2017 - Priority_Queue.thy
and mset :: "'q ⇒ 'a multiset"
assumes mset_empty: "mset empty = {#}"
and is_empty: "invar q ⟹ is_empty q = (mset q = {#})"
and mset_insert: "invar q ⟹ mset (insert x q) = mset q + {#x#}"
and mset_del_min: "invar q ⟹ mset q ≠ {#} ⟹
mset (del_min q) = mset q - {# get_min q #}"
and mset_get_min: "invar q ⟹ mset q ≠ {#} ⟹ get_min q = Min_mset (mset q)"
and invar_empty: "invar empty"
and invar_insert: "invar q ⟹ invar (insert x q)"
and invar_del_min: "invar q ⟹ mset q ≠ {#} ⟹ invar (del_min q)"
text ‹Extend locale with ‹merge›. Need to enforce that ‹'q› is the same in both locales.›
locale Priority_Queue_Merge = Priority_Queue where empty = empty for empty :: 'q +
fixes merge :: "'q ⇒ 'q ⇒ 'q"
assumes mset_merge: "⟦ invar q1; invar q2 ⟧ ⟹ mset (merge q1 q2) = mset q1 + mset q2"
locale
Priority_Queue

Isabelle2017 - ex11_tmpl.thy
orig.invar_empty orig.invar_insert orig.invar_del_min
text ‹Show that your new implementation satisfies the priority queue interface!›
sublocale Priority_Queue empty is_empty insert get_min del_min invar mset
apply unfold_locales
proof goal_cases
case 1
then show ?case sorry
next
case (2 q) -- ‹and so on›
oops
proof (state)
goal (8 subgoals):
1. local.mset local.empty = {#}
2. ⋀q. invar q ⟹ is_empty q = (local.mset q = {#})
3. ⋀q x. invar q ⟹ local.mset (local.insert x q) = local.mset q + {#x#}
4. ⋀q. ⟦invar q; local.mset q ≠ {#}⟧ ⟹ local.mset (local.del_min q) = local.mset q - {#local.get_min q#}
5. ⋀q. ⟦invar q; local.mset q ≠ {#}⟧ ⟹ local.get_min q = Min_mset (local.mset q)
6. invar local.empty

Isabelle2017 - ex11_tmpl.thy (modified)
text ‹Show that your new implementation satisfies the priority queue interface!›
sublocale Priority_Queue empty is_empty insert get_min del_min invar mset
apply unfold_locales
proof goal_cases
case 1
then show ?case
next
case (2 q) -- ‹and so on›
oops
proof (prove)
goal (1 subgoal):
1. local.mset local.empty = {#}

Isabelle2017 - ex11_tmpl.thy
apply unfold_locales
proof goal_cases
case 1
then show ?case
unfolding empty_def by simp
next
case (2 q) then show ?case
apply auto
end
proof (prove)
goal (2 subgoals):
1. ⟦invar q; is_empty q⟧ ⟹ local.mset q = {#}
2. ⟦invar q; local.mset q = {#}⟧ ⟹ is_empty q

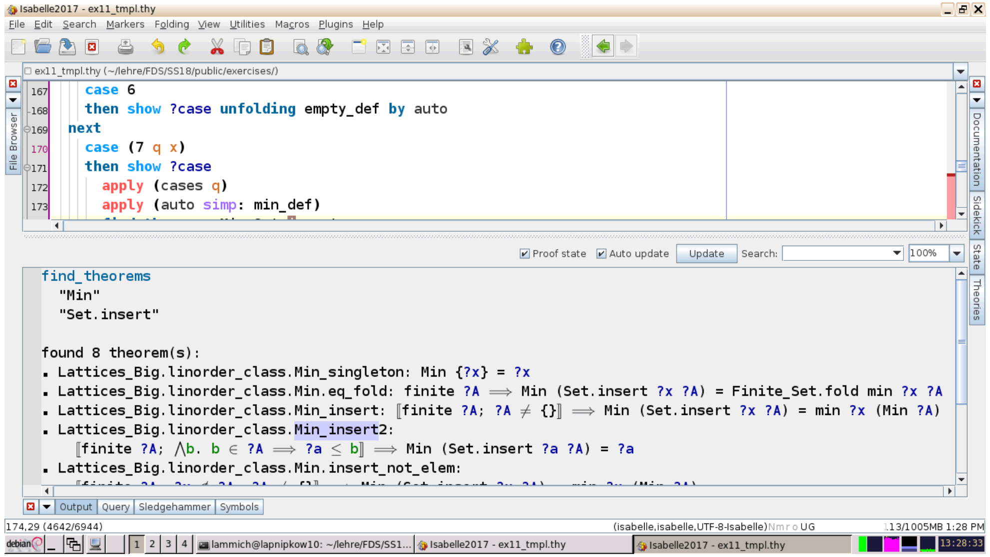
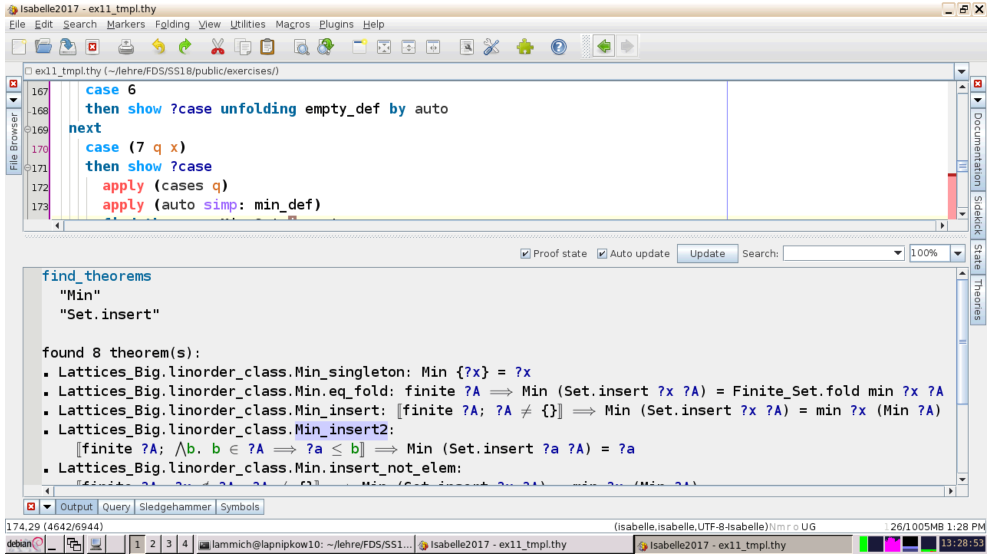
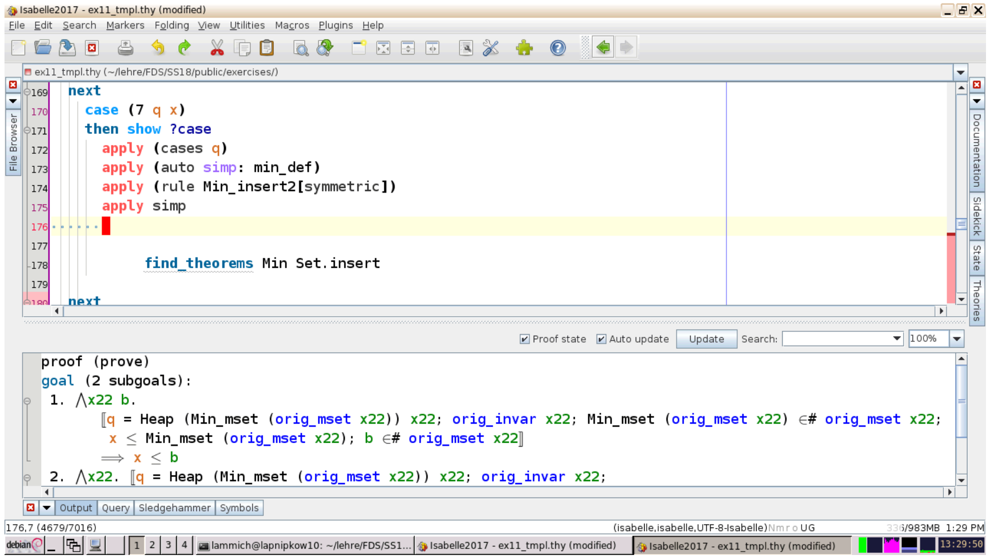
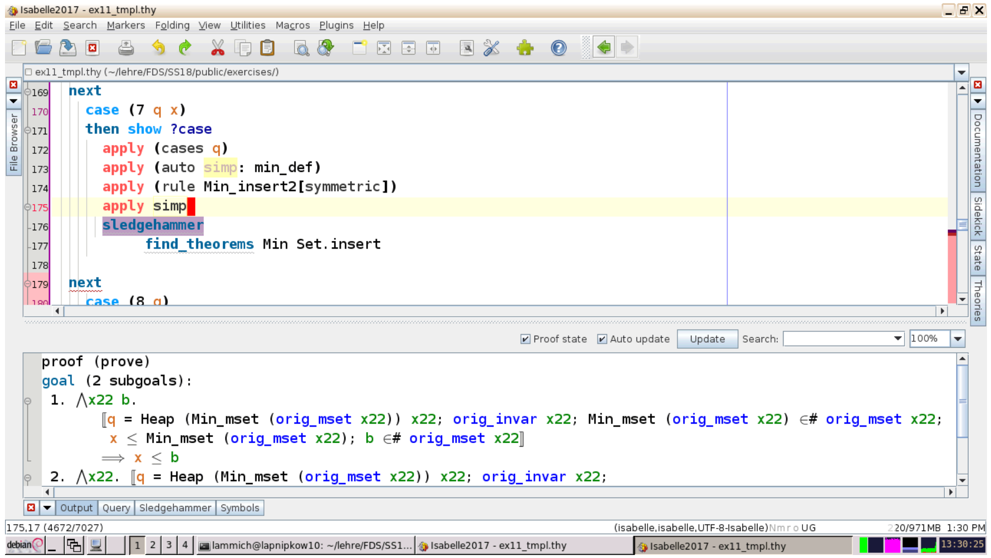

Isabelle2017 - ex11_tmpl.thy
ex11_tmpl.thy (~/lehre/FDS/SS18/public/exercises/)
apply unfold_locales
proof goal_cases
case 1
then show ?case
unfolding empty_def by simp
next
case (2 q) then show ?case
apply auto
end
proof (prove)
goal (2 subgoals):
1. ⟦invar q; is_empty q⟧ ⟹ local.mset q = {#}
2. ⟦invar q; local.mset q = {#}⟧ ⟹ is_empty q
156,17 (4259/6488)

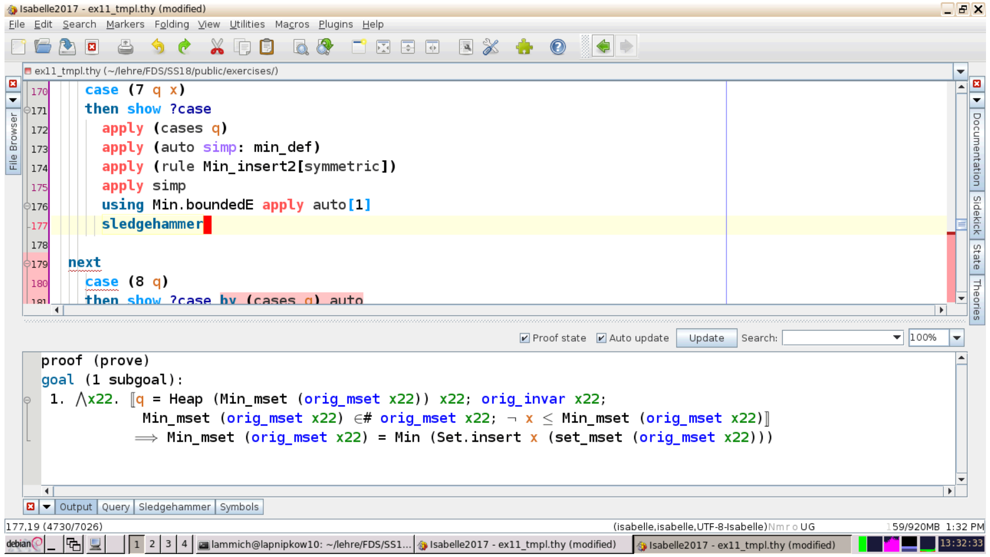

Isabelle2017 - ex11_tmpl.thy
ex11_tmpl.thy (~/lehre/FDS/SS18/public/exercises/)
"mset Empty = {#}"
| "mset (Heap _ q) = orig_mset q"
thm orig.mset_get_min
fun invar :: "('a,'q) bs_pq ⇒ bool"
where
"invar Empty ⟷ True"
| "invar (Heap a q) ⟷ orig_invar q ∧ a∈#orig_mset q ∧ a=Min_mset (orig_mset q)"
consts
invar :: "('a, 'q) bs_pq ⇒ bool"
Found termination order: "{}"
109,7 (2923/6488)

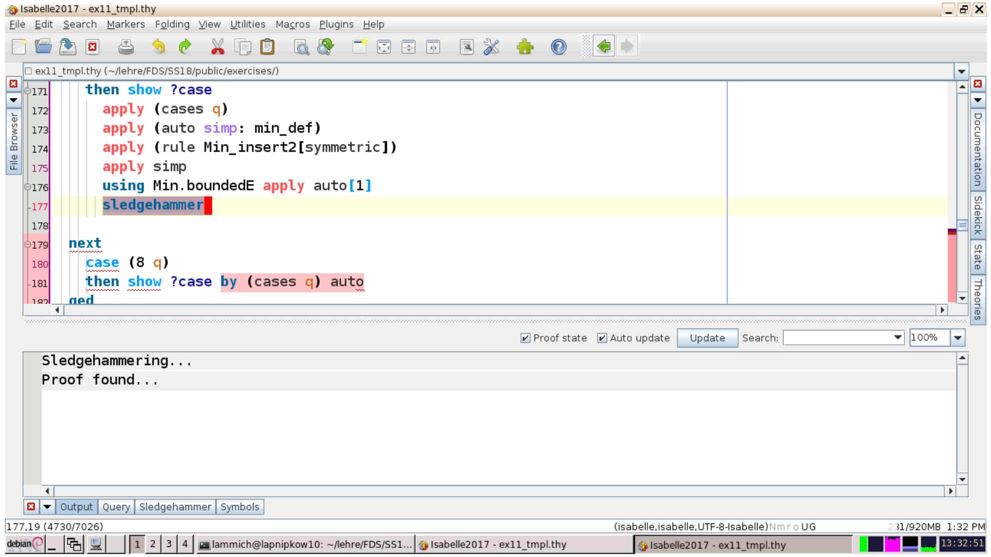

Isabelle2017 - ex11_tmpl.thy
ex11_tmpl.thy (~/lehre/FDS/SS18/public/exercises/)
proof goal_cases
case 1
then show ?case
unfolding empty_def by simp
next
case (2 q) then show ?case
apply auto
end
proof (prove)
goal (2 subgoals):
1. ⟦invar q; is_empty q⟧ ⟹ local.mset q = {#}
2. ⟦invar q; local.mset q = {#}⟧ ⟹ is_empty q
156,1 (4243/6488)

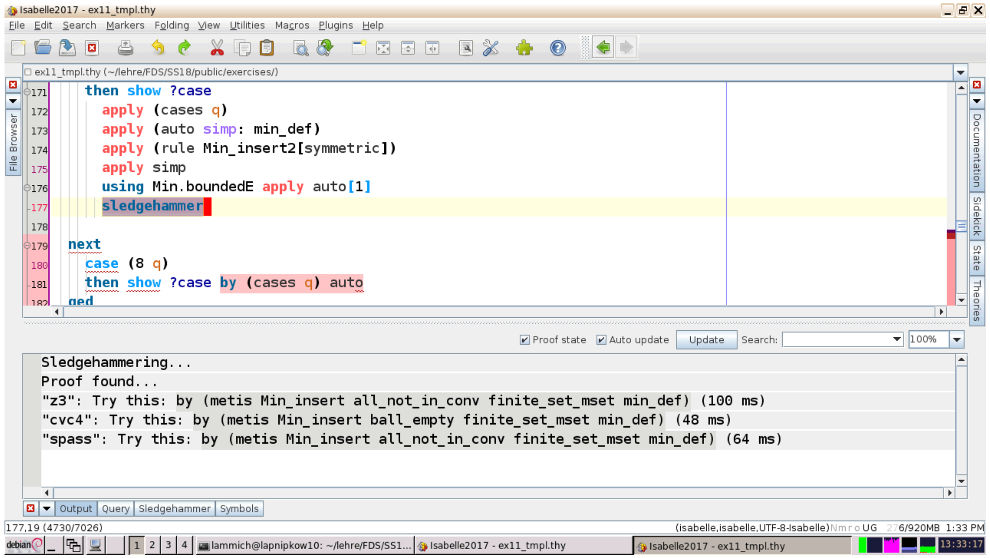

Isabelle2017 - ex11_tmpl.thy
ex11_tmpl.thy (~/lehre/FDS/SS18/public/exercises/)
next
case 6
then show ?case unfolding empty_def by auto
next
case (7 q x)
then show ?case
apply (cases q)
apply (auto )
next
case (8 q)
then show ?case by (cases q) auto
proof (prove)
using this:
invar q
goal (1 subgoal):
1. invar (local.insert x q)
171,20 (4558/6895)

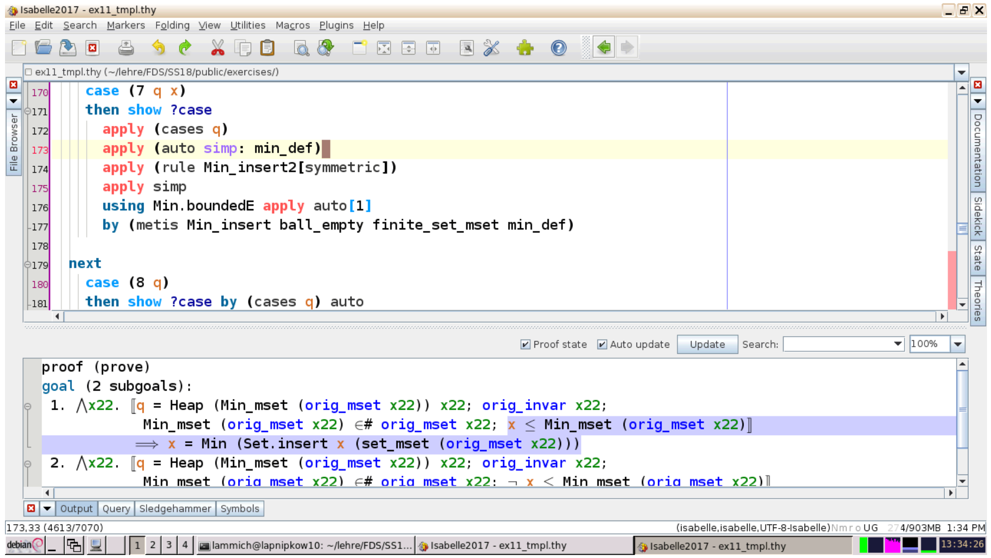

Isabelle2017 - ex11_tmpl.thy (modified)
case 6
then show ?case unfolding empty_def by auto
next
case (7 q x)
then show ?case
apply (cases q)
apply (auto simp: min_def)
next
case (8 q)
then show ?case by (cases q) auto
proof (prove)
goal (2 subgoals):
1. ⋀x22. ⟦q = Heap (Min_mset (orig_mset x22)) x22; orig_invar x22;
Min_mset (orig_mset x22) ∈# orig_mset x22; x ≤ Min_mset (orig_mset x22)⟧
⟹ x = Min (Set.insert x (set_mset (orig_mset x22)))
2. ⋀x22. ⟦q = Heap (Min_mset (orig_mset x22)) x22; orig_invar x22;
Min_mset (orig_mset x22) ∈# orig_mset x22; ¬ x ≤ Min_mset (orig_mset x22)⟧
Output Query Sledgehammer Symbols
173,32 (4612/6915) Input/output complete

Isabelle2017 - ex11_tmpl.thy (modified)
case 6
then show ?case unfolding empty_def by auto
next
case (7 q x)
then show ?case
apply (cases q)
apply (auto simp: min_def)
next
case (8 q)
then show ?case by (cases q) auto
proof (prove)
goal (2 subgoals):
1. ⋀x22. ⟦q = Heap (Min_mset (orig_mset x22)) x22; orig_invar x22;
Min_mset (orig_mset x22) ∈# orig_mset x22; x ≤ Min_mset (orig_mset x22)⟧
⟹ x = Min (Set.insert x (set_mset (orig_mset x22)))
2. ⋀x22. ⟦q = Heap (Min_mset (orig_mset x22)) x22; orig_invar x22;
Min_mset (orig_mset x22) ∈# orig_mset x22; ¬ x ≤ Min_mset (orig_mset x22)⟧
Output Query Sledgehammer Symbols
173,32 (4612/6915)

Isabelle2017 - ex11_tmpl.thy (modified)
case 6
then show ?case unfolding empty_def by auto
next
case (7 q x)
then show ?case
apply (cases q)
apply (auto simp: min_def)
next
case (8 q)
then show ?case by (cases q) auto
proof (prove)
goal (2 subgoals):
1. ⋀x22. ⟦q = Heap (Min_mset (orig_mset x22)) x22; orig_invar x22;
Min_mset (orig_mset x22) ∈# orig_mset x22; x ≤ Min_mset (orig_mset x22)⟧
⟹ x = Min (Set.insert x (set_mset (orig_mset x22)))
2. ⋀x22. ⟦q = Heap (Min_mset (orig_mset x22)) x22; orig_invar x22;
Min_mset (orig_mset x22) ∈# orig_mset x22; ¬ x ≤ Min_mset (orig_mset x22)⟧
Output Query Sledgehammer Symbols
173,33 (4613/6915)

Isabelle2017 - ex11_tmpl.thy
case 6
then show ?case unfolding empty_def by auto
next
case (7 q x)
then show ?case
apply (cases q)
apply (auto simp: min_def)
sledgehammer
next
case (8 q)
Sledgehammering...
Output Query Sledgehammer Symbols
174,19 (4632/6928)

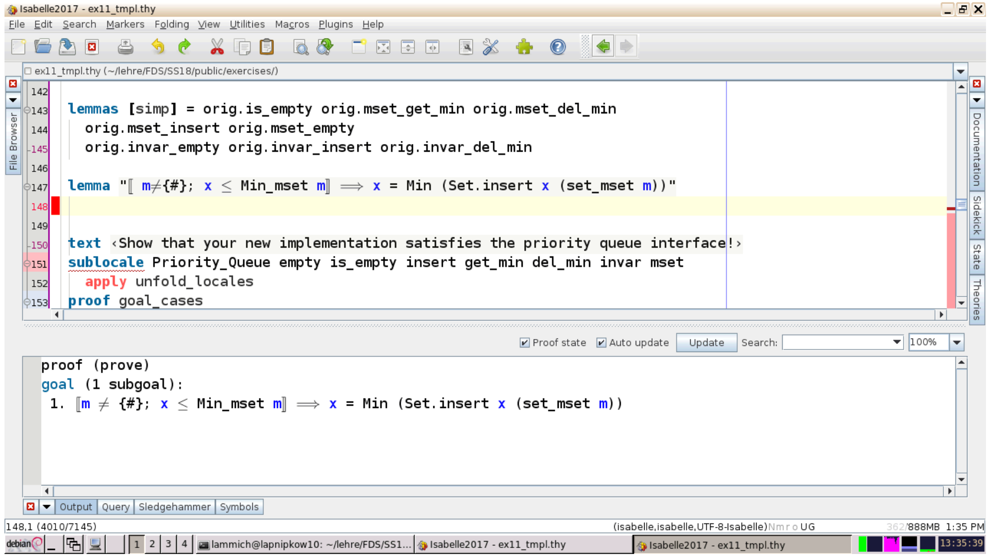
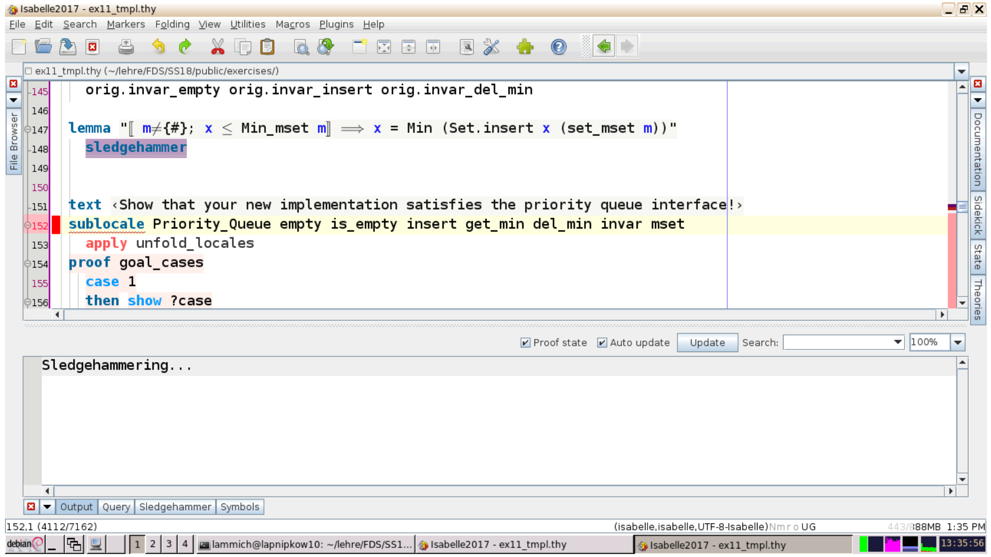
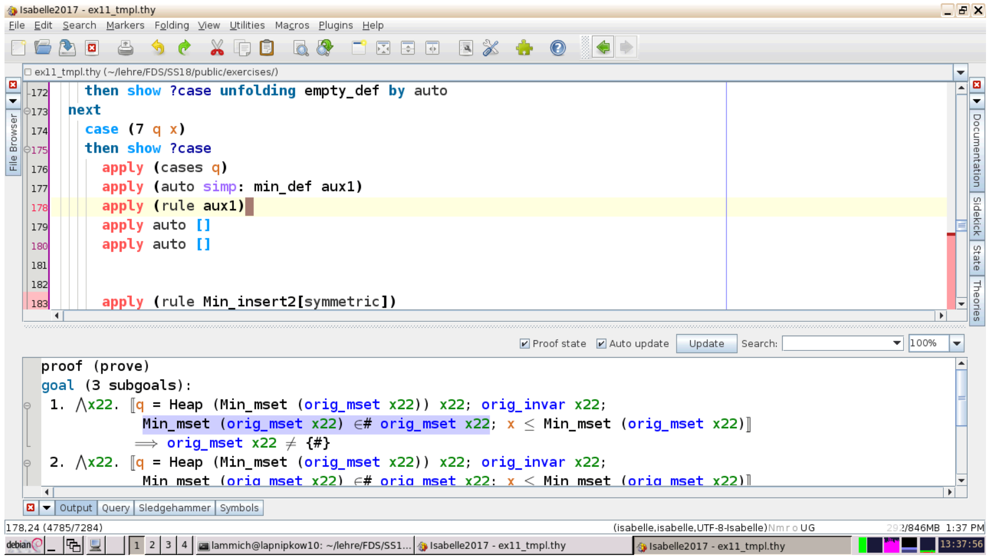
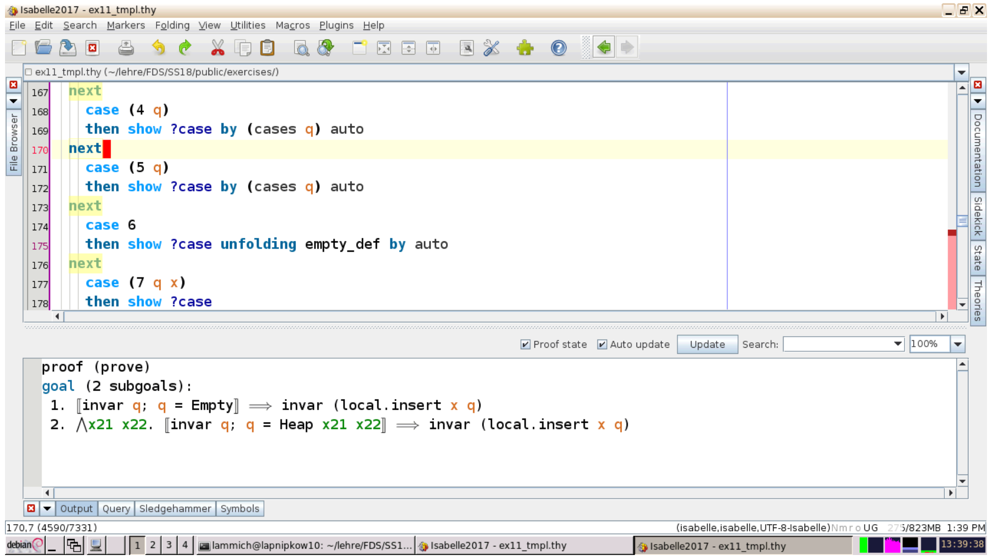

Isabelle2017 - ex11_tmpl.thy
case 6
then show ?case unfolding empty_def by auto
next
case (7 q x)
then show ?case
apply (cases q)
apply (auto simp: min_def)
find_theorems
"Min"
"Set.insert"
found 8 theorem(s):
Lattices_Big.linorder_class.Min_singleton: Min {?x} = ?x
Lattices_Big.linorder_class.Min.eq_fold: finite ?A ⟹ Min (Set.insert ?x ?A) = Finite_Set.fold min ?x ?A
Lattices_Big.linorder_class.Min_insert: ⟦finite ?A; ?A ≠ {}⟧ ⟹ Min (Set.insert ?x ?A) = min ?x (Min ?A)
Lattices_Big.linorder_class.Min_insert2:
⟦finite ?A; ⋀b. b ∈ ?A ⟹ ?a ≤ b⟧ ⟹ Min (Set.insert ?a ?A) = ?a
Lattices_Big.linorder_class.Min.insert_not_elem:

Isabelle2017 - ex11_tmpl.thy
case 6
then show ?case unfolding empty_def by auto
next
case (7 q x)
then show ?case
apply (cases q)
apply (auto simp: min_def)
find_theorems
"Min"
"Set.insert"
found 8 theorem(s):
Lattices_Big.linorder_class.Min_singleton: Min {?x} = ?x
Lattices_Big.linorder_class.Min.eq_fold: finite ?A ⟹ Min (Set.insert ?x ?A) = Finite_Set.fold min ?x ?A
Lattices_Big.linorder_class.Min_insert: ⟦finite ?A; ?A ≠ {}⟧ ⟹ Min (Set.insert ?x ?A) = min ?x (Min ?A)
Lattices_Big.linorder_class.Min_insert2:
⟦finite ?A; ⋀b. b ∈ ?A ⟹ ?a ≤ b⟧ ⟹ Min (Set.insert ?a ?A) = ?a
Lattices_Big.linorder_class.Min.insert_not_elem:

Isabelle2017 - ex11_tmpl.thy (modified)
next
case (7 q x)
then show ?case
apply (cases q)
apply (auto simp: min_def)
apply (rule Min_insert2[symmetric])
apply simp
find_theorems Min Set.insert
next
proof (prove)
goal (2 subgoals):
1. ⋀x22 b.
⟦q = Heap (Min_mset (orig_mset x22)) x22; orig_invar x22; Min_mset (orig_mset x22) ∈# orig_mset x22;
x ≤ Min_mset (orig_mset x22); b ∈# orig_mset x22⟧
⟹ x ≤ b
2. ⋀x22. ⟦q = Heap (Min_mset (orig_mset x22)) x22; orig_invar x22;

Isabelle2017 - ex11_tmpl.thy
next
case (7 q x)
then show ?case
apply (cases q)
apply (auto simp: min_def)
apply (rule Min_insert2[symmetric])
apply simp
sledgehammer
find_theorems Min Set.insert
next
case (8 q)
proof (prove)
goal (2 subgoals):
1. ⋀x22 b.
⟦q = Heap (Min_mset (orig_mset x22)) x22; orig_invar x22; Min_mset (orig_mset x22) ∈# orig_mset x22;
x ≤ Min_mset (orig_mset x22); b ∈# orig_mset x22⟧
⟹ x ≤ b
2. ⋀x22. ⟦q = Heap (Min_mset (orig_mset x22)) x22; orig_invar x22;

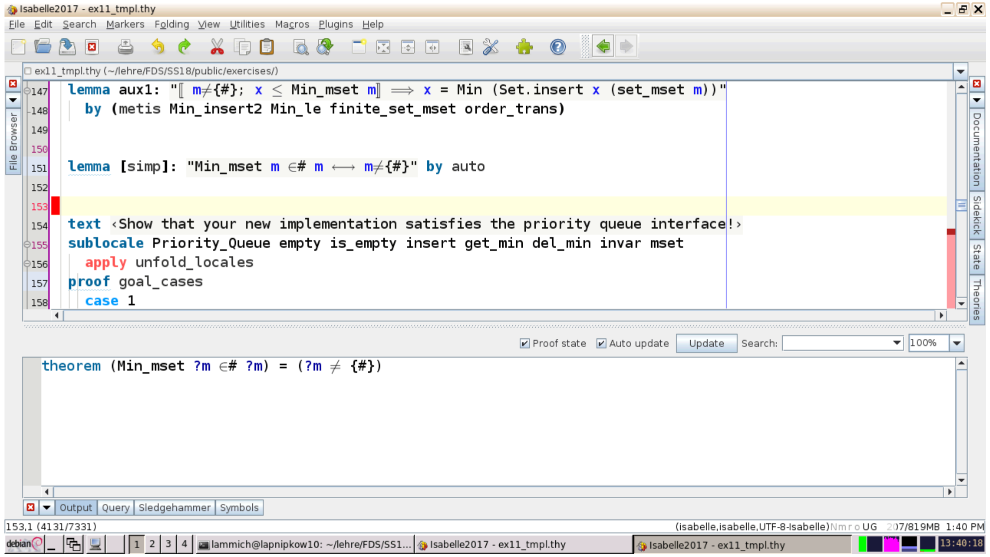

Isabelle2017 - ex11_tmpl.thy (modified)
ex11_tmpl.thy (~/lehre/FDS/SS18/public/exercises/)
case (7 q x)
then show ?case
apply (cases q)
apply (auto simp: min_def)
apply (rule Min_insert2[symmetric])
apply simp
using Min.boundedE apply auto[1]
sledgehammer
next
case (8 q)
then show ?case by (cases q) auto
proof (prove)
goal (1 subgoal):
1. ⋀x22. ⟦q = Heap (Min_mset (orig_mset x22)) x22; orig_invar x22;
Min_mset (orig_mset x22) ∈# orig_mset x22; ¬ x ≤ Min_mset (orig_mset x22)⟧
⟹ Min_mset (orig_mset x22) = Min (Set.insert x (set_mset (orig_mset x22)))
Output Query Sledgehammer Symbols

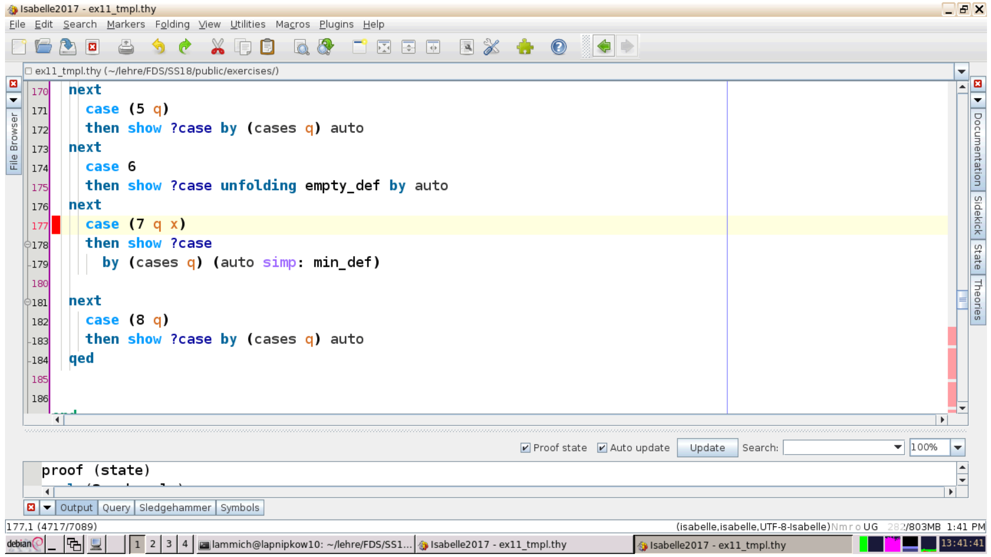

Isabelle2017 - ex11_tmpl.thy
ex11_tmpl.thy (~/lehre/FDS/SS18/public/exercises/)
then show ?case
apply (cases q)
apply (auto simp: min_def)
apply (rule Min_insert2[symmetric])
apply simp
using Min.boundedE apply auto[1]
sledgehammer
next
case (8 q)
then show ?case by (cases q) auto
qed
Sledgehammering...
Proof found...
Output Query Sledgehammer Symbols

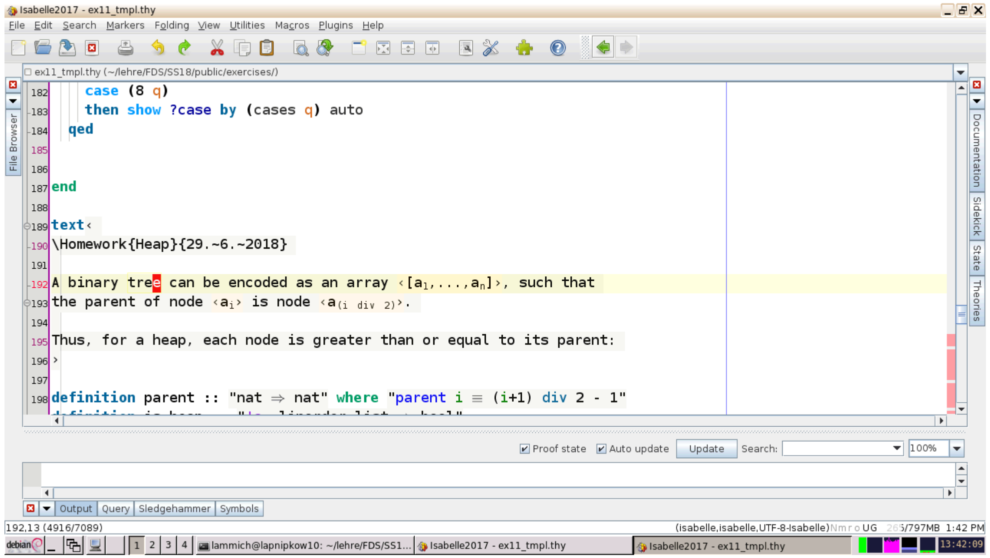

Isabelle2017 - ex11_tmpl.thy
ex11_tmpl.thy (~/lehre/FDS/SS18/public/exercises/)
then show ?case
apply (cases q)
apply (auto simp: min_def)
apply (rule Min_insert2[symmetric])
apply simp
using Min.boundedE apply auto[1]
sledgehammer
next
case (8 q)
then show ?case by (cases q) auto
qed
Sledgehammering...
Proof found...
"z3": Try this: by (metis Min_insert all_not_in_conv finite_set_mset min_def) (100 ms)
"cvc4": Try this: by (metis Min_insert ball_empty finite_set_mset min_def) (48 ms)
"spass": Try this: by (metis Min_insert all_not_in_conv finite_set_mset min_def) (64 ms)
Output Query Sledgehammer Symbols

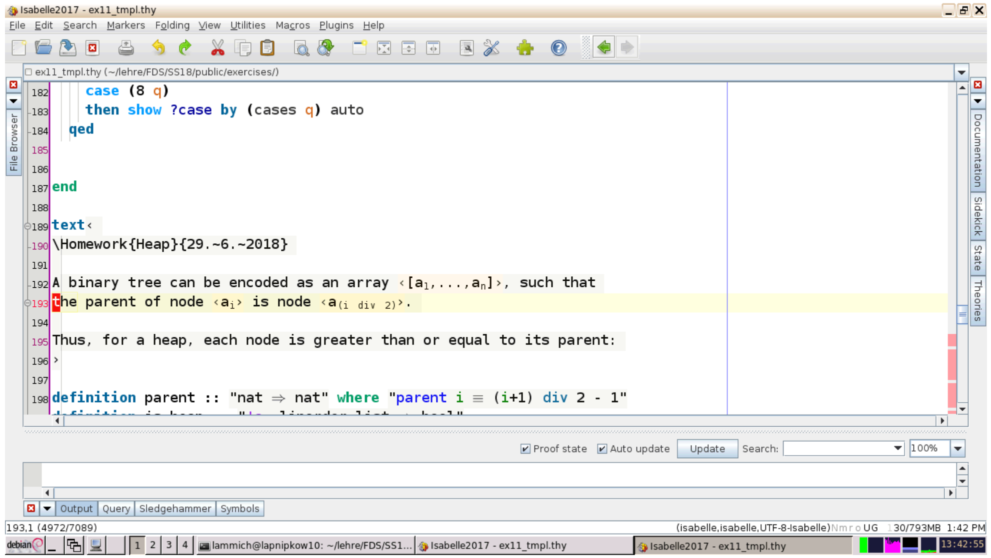

Isabelle2017 - ex11_tmpl.thy
ex11_tmpl.thy (~/lehre/FDS/SS18/public/exercises/)
case (7 q x)
then show ?case
apply (cases q)
apply (auto simp: min_def)
apply (rule Min_insert2[symmetric])
apply simp
using Min.boundedE apply auto[1]
by (metis Min_insert ball_empty finite_set_mset min_def)
next
case (8 q)
then show ?case by (cases q) auto
proof (prove)
goal (2 subgoals):
1. ⋀x22. ⟦q = Heap (Min_mset (orig_mset x22)) x22; orig_invar x22;
Min_mset (orig_mset x22) ∈# orig_mset x22; x ≤ Min_mset (orig_mset x22)⟧
⟹ x = Min (Set.insert x (set_mset (orig_mset x22)))
2. ⋀x22. ⟦q = Heap (Min_mset (orig_mset x22)) x22; orig_invar x22;
Output Query Sledgehammer Symbols

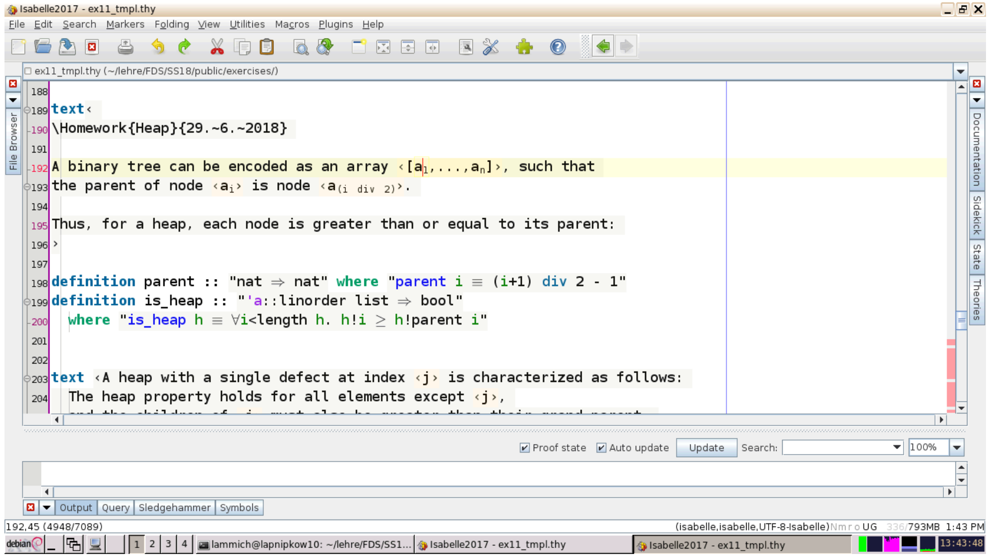

Isabelle2017 - ex11_tmpl.thy
ex11_tmpl.thy (~/lehre/FDS/SS18/public/exercises/)
lemmas [simp] = orig.is_empty orig.mset_get_min orig.mset_del_min
orig.mset_insert orig.mset_empty
orig.invar_empty orig.invar_insert orig.invar_del_min
lemma "⟦ m≠{#}; x ≤ Min_mset m⟧ ⟹ x = Min (Set.insert x (set_mset m))"
text ‹Show that your new implementation satisfies the priority queue interface!›
sublocale Priority_Queue empty is_empty insert get_min del_min invar mset
apply unfold_locales
proof goal_cases
proof (prove)
goal (1 subgoal):
1. ⟦m ≠ {#}; x ≤ Min_mset m⟧ ⟹ x = Min (Set.insert x (set_mset m))
Output Query Sledgehammer Symbols

Isabelle2017 - ex11_tmpl.thy
ex11_tmpl.thy (~/lehre/FDS/SS18/public/exercises/)
orig.invar_empty orig.invar_insert orig.invar_del_min
lemma "⟦ m≠{#}; x ≤ Min_mset m⟧ ⟹ x = Min (Set.insert x (set_mset m))"
sledgehammer
text ‹Show that your new implementation satisfies the priority queue interface!›
sublocale Priority_Queue empty is_empty insert get_min del_min invar mset
apply unfold_locales
proof goal_cases
case 1
then show ?case
Sledgehammering...
Output Query Sledgehammer Symbols

Isabelle2017 - ex11_tmpl.thy
ex11_tmpl.thy (~/lehre/FDS/SS18/public/exercises/)
then show ?case unfolding empty_def by auto
next
case (7 q x)
then show ?case
apply (cases q)
apply (auto simp: min_def aux1)
apply (rule aux1)
apply auto []
apply auto []
apply (rule Min_insert2[symmetric])
proof (prove)
goal (3 subgoals):
1. ⋀x22. ⟦q = Heap (Min_mset (orig_mset x22)) x22; orig_invar x22;
Min_mset (orig_mset x22) ∈# orig_mset x22; x ≤ Min_mset (orig_mset x22)⟧
⟹ orig_mset x22 ≠ {#}
2. ⋀x22. ⟦q = Heap (Min_mset (orig_mset x22)) x22; orig_invar x22;
Min_mset (orig_mset x22) ∈# orig_mset x22; x ≤ Min_mset (orig_mset x22)⟧
Output Query Sledgehammer Symbols

Isabelle2017 - ex11_tmpl.thy
ex11_tmpl.thy (~/lehre/FDS/SS18/public/exercises/)
next
case (4 q)
then show ?case by (cases q) auto
next
case (5 q)
then show ?case by (cases q) auto
next
case 6
then show ?case unfolding empty_def by auto
next
case (7 q x)
then show ?case
proof (prove)
goal (2 subgoals):
1. ⟦invar q; q = Empty⟧ ⟹ invar (local.insert x q)
2. ⋀x21 x22. ⟦invar q; q = Heap x21 x22⟧ ⟹ invar (local.insert x q)
Output Query Sledgehammer Symbols


Isabelle2017 - ex11_tmpl.thy
lemma aux1: "⟦ m≠{#}; x ≤ Min_mset m⟧ ⟹ x = Min (Set.insert x (set_mset m))"
by (metis Min_insert2 Min_le finite_set_mset order_trans)
lemma [simp]: "Min_mset m ∈# m ⟷ m≠{#}" by auto
text ‹Show that your new implementation satisfies the priority queue interface!›
sublocale Priority_Queue empty is_empty insert get_min del_min invar mset
apply unfold_locales
proof goal_cases
case 1
theorem (Min_mset ?m ∈# ?m) = (?m ≠ {#})

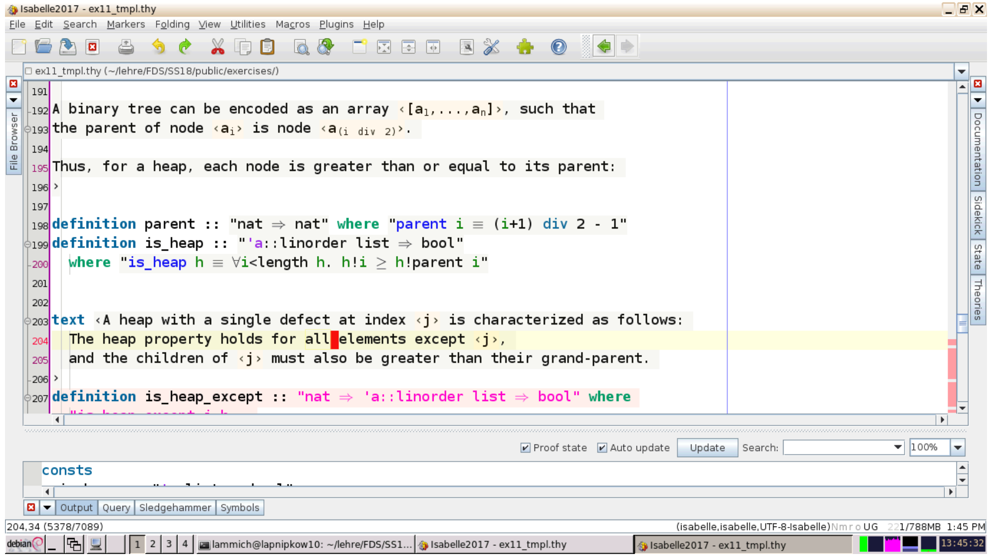

Isabelle2017 - ex11_tmpl.thy
next
case (5 q)
then show ?case by (cases q) auto
next
case 6
then show ?case unfolding empty_def by auto
next
case (7 q x)
then show ?case
by (cases q) (auto simp: min_def)
next
case (8 q)
then show ?case by (cases q) auto
qed
proof (state)


Isabelle2017 - ex11_tmpl.thy
case (8 q)
then show ?case by (cases q) auto
qed
end
text‹
\Homework{Heap}{29.~6.~2018}
A binary tree can be encoded as an array ‹[a₁,...,aₙ]›, such that
the parent of node ‹aᵢ› is node ‹a(i div 2)›.
Thus, for a heap, each node is greater than or equal to its parent:
›
definition parent :: "nat ⇒ nat" where "parent i ≡ (i+1) div 2 - 1"

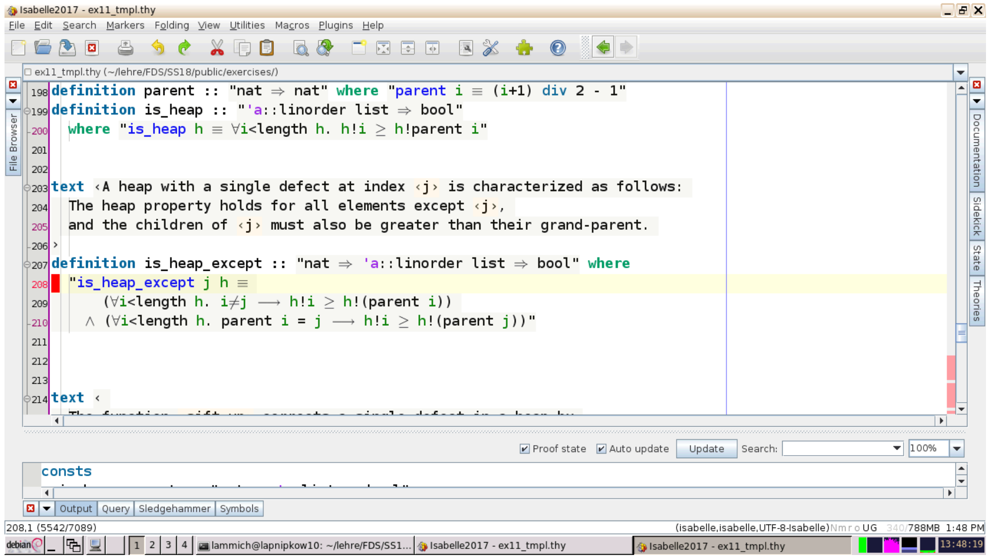

Isabelle2017 - ex11_tmpl.thy
case (8 q)
then show ?case by (cases q) auto
qed
end
text‹
\Homework{Heap}{29.~6.~2018}
A binary tree can be encoded as an array ‹[a₁,...,aₙ]›, such that
the parent of node ‹aᵢ› is node ‹a(i div 2)›.
Thus, for a heap, each node is greater than or equal to its parent:
›
definition parent :: "nat ⇒ nat" where "parent i ≡ (i+1) div 2 - 1"



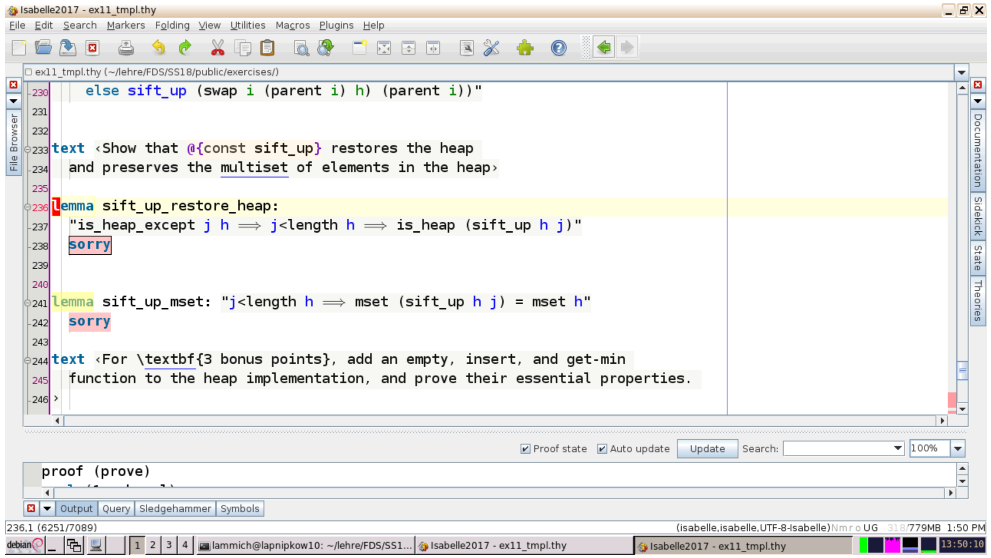
Isabelle2017 - ex11_tmpl.thy
text ‹
\Homework{Heap}{29.~6.~2018}
A binary tree can be encoded as an array ‹[a1,...,an]›, such that
the parent of node ‹ai› is node ‹a(i div 2)›.
Thus, for a heap, each node is greater than or equal to its parent:
›
definition parent :: "nat ⇒ nat" where "parent i ≡ (i+1) div 2 - 1"
definition is_heap :: "'a::linorder list ⇒ bool"
where "is_heap h ≡ ∀i<length h. h!i ≥ h!parent i"
text ‹A heap with a single defect at index ‹j› is characterized as follows:
The heap property holds for all elements except ‹j›,



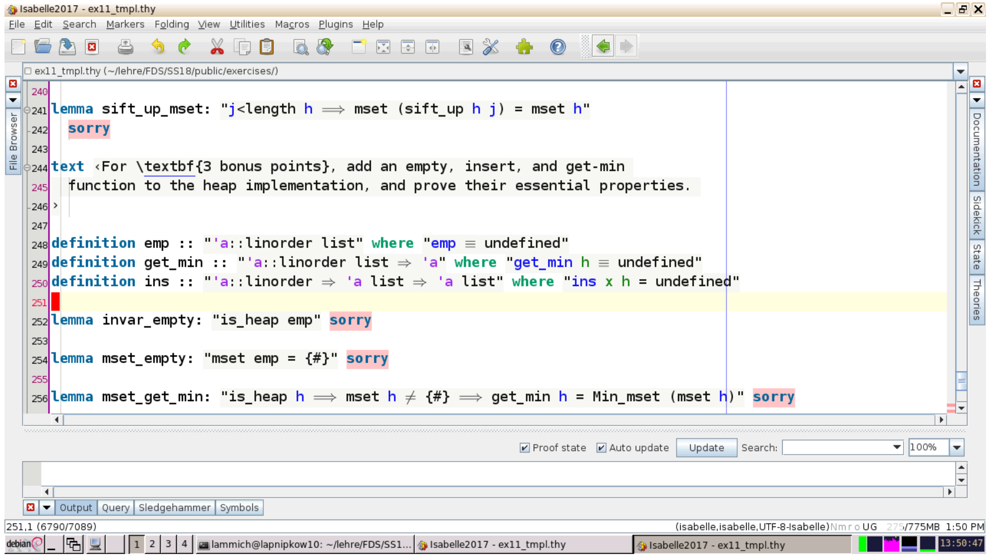
Isabelle2017 - ex11_tmpl.thy
A binary tree can be encoded as an array ‹[a1,...,an]›, such that
the parent of node ‹ai› is node ‹a(i div 2)›.
Thus, for a heap, each node is greater than or equal to its parent:
›
definition parent :: "nat ⇒ nat" where "parent i ≡ (i+1) div 2 - 1"
definition is_heap :: "'a::linorder list ⇒ bool"
where "is_heap h ≡ ∀i<length h. h!i ≥ h!parent i"
text ‹A heap with a single defect at index ‹j› is characterized as follows:
The heap property holds for all elements except ‹j›,
and the children of ‹j› must also be greater than their grand-parent.
›
definition is_heap_except :: "nat ⇒ 'a::linorder list ⇒ bool" where
consts

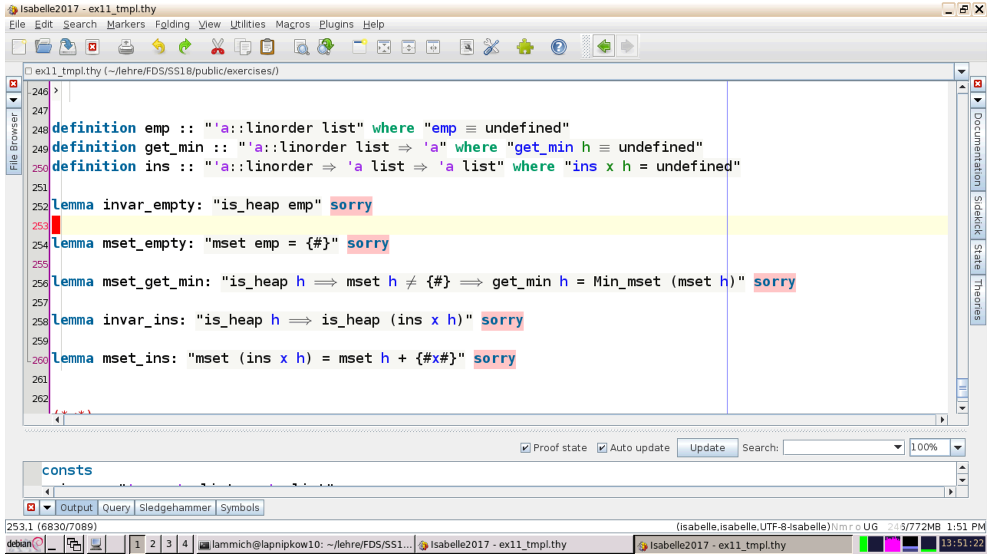


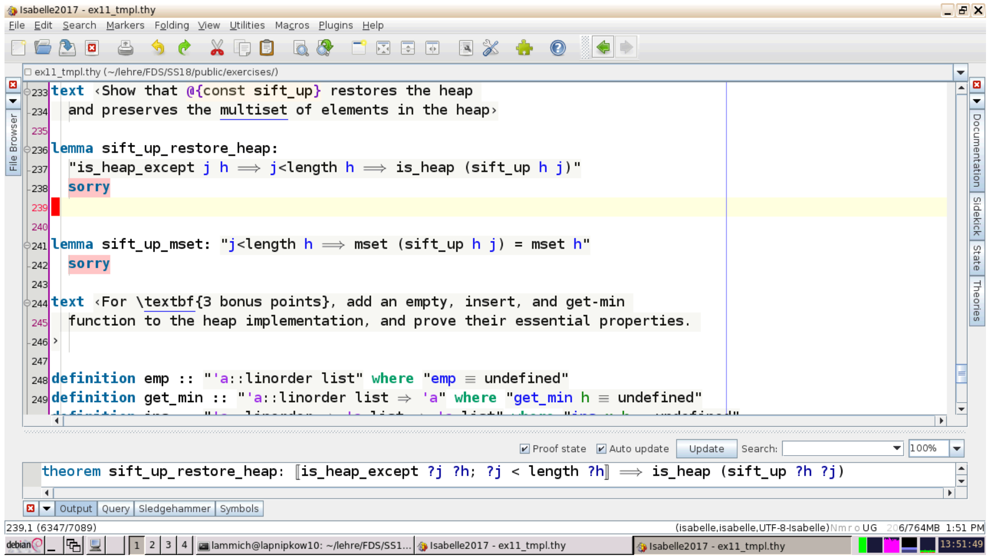
Isabelle2017 - ex11_tmpl.thy
definition parent :: "nat ⇒ nat" where "parent i ≡ (i+1) div 2 - 1"
definition is_heap :: "'a::linorder list ⇒ bool"
where "is_heap h ≡ ∀i<length h. h!i ≥ h!parent i"
text ‹A heap with a single defect at index ‹j› is characterized as follows:
The heap property holds for all elements except ‹j›,
and the children of ‹j› must also be greater than their grand-parent.
›
definition is_heap_except :: "nat ⇒ 'a::linorder list ⇒ bool" where
"is_heap_except j h ≡
(∀i<length h. i≠j ⟶ h!i ≥ h!(parent i))
∧ (∀i<length h. parent i = j ⟶ h!i ≥ h!(parent j))"
text ‹
consts



Isabelle2017 - ex11_tmpl.thy
else sift_up (swap i (parent i) h) (parent i))"
text ‹Show that @{const sift_up} restores the heap
and preserves the multiset of elements in the heap›
lemma sift_up_restore_heap:
"is_heap_except j h ⟹ j<length h ⟹ is_heap (sift_up h j)"
sorry
lemma sift_up_mset: "j<length h ⟹ mset (sift_up h j) = mset h"
sorry
text ‹For \textbf{3 bonus points}, add an empty, insert, and get-min
function to the heap implementation, and prove their essential properties.
›
proof (prove)


Isabelle2017 - ex11_tmpl.thy
ex11_tmpl.thy (~/lehre/FDS/SS18/public/exercises/)
lemma sift_up_mset: "j<length h ⟹ mset (sift_up h j) = mset h"
sorry
text ‹For \textbf{3 bonus points}, add an empty, insert, and get-min
function to the heap implementation, and prove their essential properties.
›
definition emp :: "'a::linorder list" where "emp ≡ undefined"
definition get_min :: "'a::linorder list ⇒ 'a" where "get_min h ≡ undefined"
definition ins :: "'a::linorder ⇒ 'a list ⇒ 'a list" where "ins x h = undefined"
lemma invar_empty: "is_heap emp" sorry
lemma mset_empty: "mset emp = {#}" sorry
lemma mset_get_min: "is_heap h ⟹ mset h ≠ {#} ⟹ get_min h = Min_mset (mset h)" sorry
Proof state
Auto update
Update
Search:
100%
Output
Query
Sledgehammer
Symbols
251.1 (6790/7089)
1:50 PM


Isabelle2017 - ex11_tmpl.thy
ex11_tmpl.thy (~/lehre/FDS/SS18/public/exercises/)
definition emp :: "'a::linorder list" where "emp ≡ undefined"
definition get_min :: "'a::linorder list ⇒ 'a" where "get_min h ≡ undefined"
definition ins :: "'a::linorder ⇒ 'a list ⇒ 'a list" where "ins x h = undefined"
lemma invar_empty: "is_heap emp" sorry
lemma mset_empty: "mset emp = {#}" sorry
lemma mset_get_min: "is_heap h ⟹ mset h ≠ {#} ⟹ get_min h = Min_mset (mset h)" sorry
lemma invar_ins: "is_heap h ⟹ is_heap (ins x h)" sorry
lemma mset_ins: "mset (ins x h) = mset h + {#x#}" sorry
Proof state
Auto update
Update
Search:
100%
consts
Output
Query
Sledgehammer
Symbols
253.1 (6830/7089)
1:51 PM


Isabelle2017 - ex11_tmpl.thy
ex11_tmpl.thy (~/lehre/FDS/SS18/public/exercises/)
text ‹Show that @{const sift_up} restores the heap
and preserves the multiset of elements in the heap›
lemma sift_up_restore_heap:
"is_heap_except j h ⟹ j<length h ⟹ is_heap (sift_up h j)"
sorry
lemma sift_up_mset: "j<length h ⟹ mset (sift_up h j) = mset h"
sorry
text ‹For \textbf{3 bonus points}, add an empty, insert, and get-min
function to the heap implementation, and prove their essential properties.
›
definition emp :: "'a::linorder list" where "emp ≡ undefined"
definition get_min :: "'a::linorder list ⇒ 'a" where "get_min h ≡ undefined"
Proof state
Auto update
Update
Search:
100%
theorem sift_up_restore_heap: ⟦is_heap_except ?j ?h; ?j < length ?h⟧ ⟹ is_heap (sift_up ?h ?j)
Output
Query
Sledgehammer
Symbols
239.1 (6347/7089)
1:51 PM