

Homework 5

Problem 5.1

Solution:

I will use definition 10.10 from the lecture notes, which shows the criteria for a formula to be a class invariant. Stack's class invariant formula from lecture notes is $length(elements) == size$. This is a strong class invariant. Reason is, it is always true for new instances of the class and holds before and after any public functions from the class are called.

Problem 5.2

Solution:

I will use 'top' and 'bottom' which are head and tail of the queue.

```
fun insert(x:int){
  if(data->top == Nil){
    data->top = x
  }
  else{
    *cursor = data->bottom
    *e = data->top
    while(e != Nil && *e >= x){
      e = e.tail
      cursor = cursor.tail
    }
  }
  backwardShifting(cursor) //I did not elaborate on this function
  //but it basically shifts each element in the queue one place backwards

  e=x
}
```

Problem 5.3

Solution:

Using induction:

Base Case:

$$zero + m == m + zero$$

$$m == m$$

Step Case:

Inductive Hypothesis: $n + m == m + n$

Prove for $suc(n)$:

$$suc(n) + m == m + suc(n)$$

$$suc(n + m) == m + suc(n)$$

Using Inductive Hypothesis:

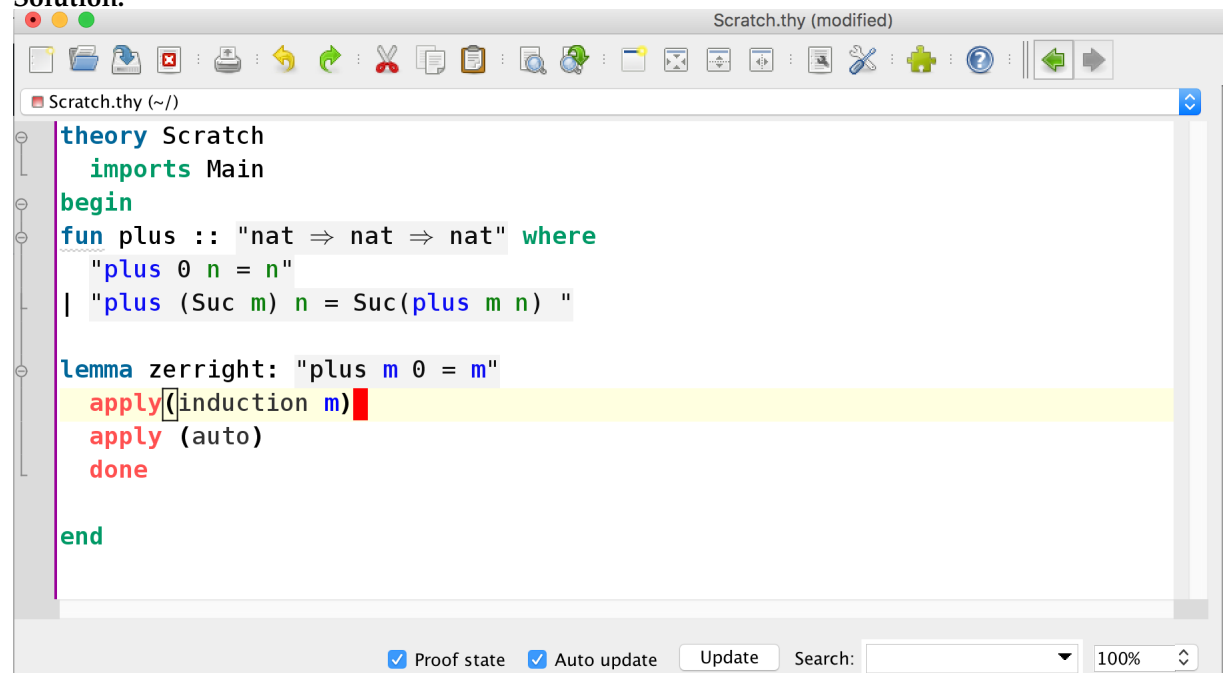
$$suc(m + n) == m + suc(n)$$

$$m + suc(n) == m + suc(n)$$

Hence, proved.

Problem 5.4

Solution:

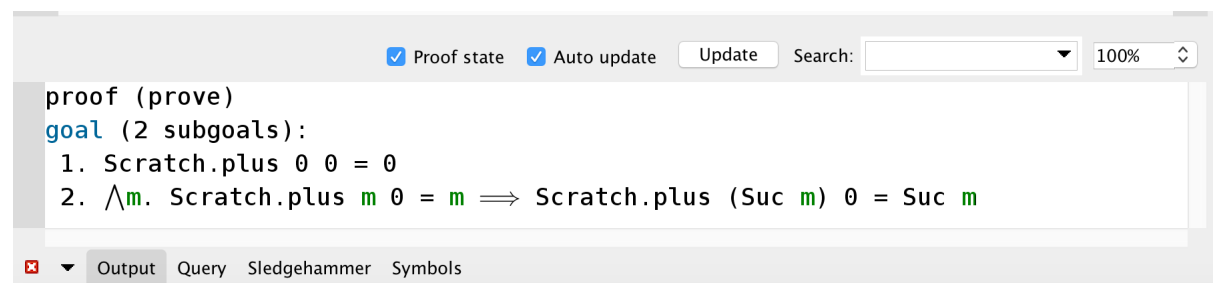


```
theory Scratch
  imports Main
begin
fun plus :: "nat  $\Rightarrow$  nat  $\Rightarrow$  nat" where
  "plus 0 n = n"
| "plus (Suc m) n = Suc (plus m n) "

lemma zerright: "plus m 0 = m"
  apply (induction m)
  apply (auto)
  done

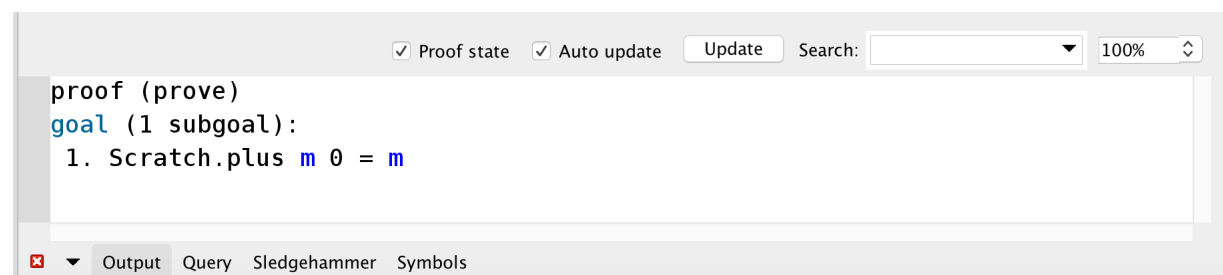
end
```

Proof state: ☒ Auto update: ☒ Update Search: 100%



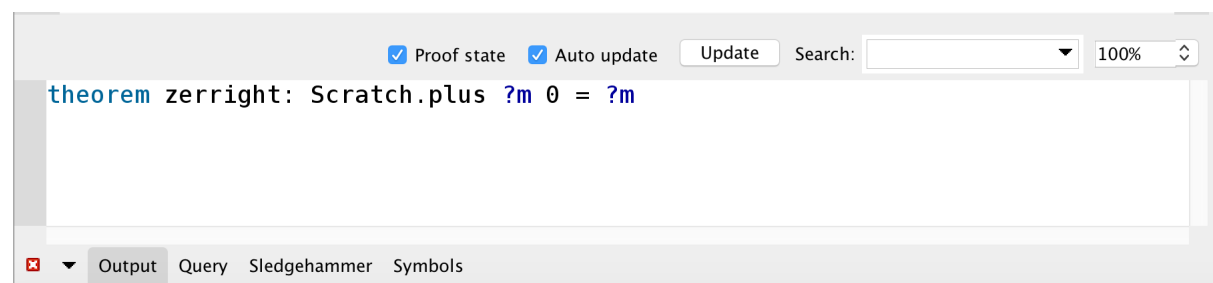
```
proof (prove)
goal (2 subgoals):
  1. Scratch.plus 0 0 = 0
  2.  $\wedge m$ . Scratch.plus m 0 = m  $\implies$  Scratch.plus (Suc m) 0 = Suc m
```

Output Query Sledgehammer Symbols



```
proof (prove)
goal (1 subgoal):
  1. Scratch.plus m 0 = m
```

Output Query Sledgehammer Symbols



```
theorem zerright: Scratch.plus ?m 0 = ?m
```

Proof state: ☒ Auto update: ☒ Update Search: 100%

Output Query Sledgehammer Symbols