Solution
Sheet 2

Exercise 2

Specification of natural numbers

```
sort Nat;

   operations

      zero : -> Nat;

      succ : Nat -> Nat;

      add : Nat x Nat -> Nat;

   declare i, j : Z;

   axioms

      add(zero,i) == i; /* A1*/

      add(succ(i),j) == succ(add(i,j)); /* A2 */
            i == add(i,zero); /* A3, proved in lecture */
```

Proof

Theorem

`add(i,j) == add(j,i)`

Proof

Induction start: `i == zero`
To demonstrate: `add(zero,j) == add(j,zero)`

`add(zero,j)` == (A1)
`j` == (A3)
`add(j,zero)`

Induction step:
Assumption: `add(i,j) == add(j,i)`
To demonstrate: `add(succ(i),j) == add(j,succ(i))`

`add(succ(i),j)` == (A2)
`succ(add(i,j))` == (Assumption)
`succ(add(j,i))` == (A2)
`add(succ(j),i)` == (A4, see below)
`add(j,succ(i))`

Auxiliary proposition A4

Theorem

```
add(succ(i),j) == add(i,succ(j))
```

Proof

Induction start: `i == zero`

To demonstrate: `add(succ(zero),j) == add(zero,succ(j))`

```
add(succ(zero),j) == (A2)
succ(add(zero,j)) == (A1)
succ(j) == (A1)
add(zero,succ(j))
```

Induction step:

Assumption: `add(succ(i),j) == add(i,succ(j))`

To demonstrate: `add(succ(succ(i)),j) == add(succ(i),succ(j))`

```
add(succ(succ(i)),j) == (A2)
succ(add(succ(i),j)) == (Assumption)
succ(add(i,succ(j))) == (A2)
add(succ(i),succ(j))
```

Exercise 3

Algebraic specification of queues

```
module Queue;
   import Bool, true, false from Bool;
      Nat, zero from Nat;
   export all;
   sort Queue;
   operations
      newqueue : -> Queue;
      enqueue : Queue x Nat -> Queue;
      isempty : Queue -> Bool;
      dequeue : Queue -> Queue;
      head : Queue -> Nat;
      tail: Queue -> Nat;
   declare q : Queue; n,m : Nat;
   axioms
      isempty(newqueue) == true;
      isempty(enqueue(q,n)) == false;
      dequeue(newqueue) == newqueue;
      dequeue(enqueue(newqueue,n)) == newqueue;
      dequeue(enqueue(enqueue(q,m),n)) == enqueue(dequeue(enqueue(q,m)),n);
      tail(newqueue) == zero;
      tail(enqueue(q,n)) == n;
      head(newqueue) == zero;
      head(enqueue(newqueue,n)) == n;
      head(enqueue(enqueue(q,m),n)) == head(enqueue(q,m));
end module Queue;
```