Lab 4 Software Specification and Testing

```
> module Lab4 where
>
> import Data.List
> import System.Random
> import Test.QuickCheck
```

---

1. Read or reread Chapter 4 of *The Haskell Road*, and make a list of questions on specific points that cause difficulty of understanding.

(Deliverables: list of questions, indication of time spent.)

---

2. Implement a random data generator for the datatype `Set Int`, where `Set` is as defined in SetOrd.hs. First do this from scratch, next give a version that uses *QuickCheck* to random test this datatype.

(Deliverables: two random test generators, indication of time spent.)

---

3. Implement operations for set intersection, set union and set difference, for the datatype `Set` defined in SetOrd.hs. Next, use automated testing to check that your implementation is correct. First use your own generator, next use QuickCheck.

(Deliverables: implementations, test properties, short test report, indication of time spent.)

---

4. Read or reread Chapter 5 of *The Haskell Road*, and make a list of questions on specific points that cause difficulty of understanding.

(Deliverables: list of questions, indication of time spent.)

---

5. Suppose we implement binary relations as list of pairs, Haskell type `[(a,a)]`. Assume the following definition:

```
> type Rel a = [(a,a)]
```

Use this to implement a function

```
  symClos :: Ord a => Rel a -> Rel a
```

that gives the symmetric closure of a relation, where the relation is represented as an ordered list of pairs. E.g., `symClos [(1,2),(2,3),(3,4)]` should give `[(1,2),(2,1),(2,3),(3,2),(3,4),(4,3)]`.

(Deliverable: Haskell program, indication of time spent.)

---

6. Use the datatype for relations from the previous exercise, plus

```
> infixr 5 @@
>
> (@@) :: Eq a => Rel a -> Rel a -> Rel a
> r @@ s =
>   nub [ (x,z) | (x,y) <- r, (w,z) <- s, y == w ]
```

to define a function

```
 trClos :: Ord a => Rel a -> Rel a
```

that gives the transitive closure of a relation, represented as an ordered list of pairs. E.g., `trClos [(1,2),(2,3),(3,4)]` should give `[(1,2),(1,3),(1,4),(2,3),(2,4),(3,4)]`.

(Deliverable: Haskell program, indication of time spent.)

---

7. Test the functions `symClos` and `trClos` from the previous exercises. Devise your own test method for this. Try to use random test generation. Define reasonable properties to test. Can you use QuickCheck? How?

(Deliverables: test code, short test report, indication of time spent.)

---

8. Is there a difference between the symmetric closure of the transitive closure of a relation R and the transitive closure of the symmetric closure of R?

Deliverable: If your answer is that these are the same, you should give an argument, if you think these are different you should give an example that illustrates the difference.

---

9. **Bonus** In the lecture notes, `Statement` is in class `Show`, but the `show` function for it is a bit clumsy. Write your own `show` function for imperative programs. Next, write a `read` function, and use `show` and `read` to state some abstract test properties for how these functions should behave. Next, use `QuickCheck` to test your implementations.

Deliverable: implementation, `QuickCheck` properties, test report.

---

10. **Extra Bonus** If this was all easy for you, you might wish to throw in a solution to a difficult problem from Project Euler.

Submission deadline is Sunday evening, October 1st, at 6 pm.