Exercises

Getting Started

---

1. QuickCheck installation
   Download eqc.zip, for example from quviq-licencer.com/downloads/eqc.zip, and unpack it. You should see a directory such as Quviq QuickCheck version 1.25.1. Follow the instructions in the README file therein.

   If you are installing a special course version of QuickCheck, then there is no need to register a licence identifier.

2. Check QuickCheck installation.
   The distribution provides a directory eqc-1…. with subdirectories *ebin*, *include* and *doc*.

   Open an Erlang shell and run eqc:start().
   QuickCheck should start.

3. Replay the examples shown in class.

4. Implement the following properties:
   a. if an element is not in a list, then deleting that element leaves the list unchanged.
   b. if an element is in a list, then deleting that element shortens the length of the list by one.'

5. Check whether the following property holds:
   Adding an element to a list and deleting it again returns the original list.

Exercises

Basic Generators

---

6. Replay the examples shown in class

7. Test the function **`last_day_of_the_month(Year, Month) -> int()`**
   Don't forget to use eqc:collect to obtain information on what you have checked.

8. Create a calendar_time() generator that generates a tuple with hour, minute and seconds.
   Use the calendar_time() generator to check whether the function time_to_seconds(Time) -> Seconds computes the right value.

9. If you are east of GMT (which Stockholm is), then local time may be ahead of universal time. Simply type calendar:universal_time() in the shell and compare with calendar_local_time().

   Check the function universal_time_to_local_time(DateTime) by checking that the local time is always ahead of the universal time, i.e., calendar:universal_time_to_local_time(DateTime) > DateTime.

   Note 1: the documentation warns that the function is only defined for 1970 and later.
   Note 2: the function slave:start can come in useful when you want to test this function on a separate node. You can use distributed nodes by starting Erlang with the –sname NodeName flag.

10. Write a generator for the dictionary data type of the dict.erl module. Test as many functions in the dict.erl module as possible. (Hint: you may choose to use `from_list` since recursive generators are possible, but a bit harder to define).

11. QuickCheck the module filename.erl in stdlib, i.e., write generators for the data types filename and components and write a property for every function in the module.