CS 476 Homework #2 Due 10:45 am on 9/8

Note: Answers to the exercises listed below should be emailed to Nishant Rodrigues (`nishant2@illinois.edu`) in *typewritten form* (latex formatting preferred) by the deadline mentioned above.

1. Give your solution to the following elementary set theory exercises in *Set Theory and Algebra in Computer Science – A Gentle Introduction to Mathematical Modeling.*

   - 54
   - 55
   - 57
   - 61

2. Give your solution to **Exercise** 2.2 in Lecture 2 as a Maude functional module containing correct definitions of all the functions mentioned in that exercise. You can use `NAT-LIST-II` in pg. 15 of Lecture 2 (you can just copy and paste it in your file) as a basis for defining the above functions by importing it (which itself imports `NATURAL` in page 14) into the function module you define using the `protecting` importation keyword. You should:

   - include some test cases for each function you define and execute such test cases with the `red` command;
   - include the text of your module and the results of evaluating the test cases in your homework solution;
   - email the file containing your module and test cases to Nishant Rodrigues (`nishant2@illinois.edu`).

Note1. Maude automatically imports the `BOOL` module as a submodule of any other functional module, unless the user explicitly disables this importation. As explained in Section 9.1 of "All About Maude," when you import the `BOOL` module, you also get for free the `if-then-else-fi` operator. Using `if-then-else-fi` can make the definition of some of the functions in **Exercise** 2.2 easier.

Note2. The functions `max` and `min` do not make sense for empty lists. Therefore, you should define them on the subsort `NeList`. This is a *good example* of why subsorts are very useful. Note that, to define `max` and `min`, you will need to define a "less than or equal" predicate on naturals as an auxiliary function.

Note3. Notice that, to define the function `odd.even` you will first have to define a new sort `Pair` whose elements are pairs of lists of natural numbers. Since Maude's syntax is *user-definable*, you can choose any syntax you like to define a constructor for pairs of lists, provided the sort `Pair` so defined does indeed represent pairs of lists. Note that, to define `get.even` and `odd.even` you will need to define also predicates `odd` and `even` on natural numbers as auxiliary functions.