CS 476 Homework #3 Due 10:45am on 2/23

Note: Answers to the exercises listed below should be handed to the instructor *in hardcopy* and in *typewritten form* (latex formatting preferred) by the deadline mentioned above. You should also email to (`meseguer@cs.uiuc.edu`) the Maude code for the exercises requiring that.

1. Consider the following module, that defines the usual strict order `_<_` relation on natural numbers, and a `min` function for computing the smallest element of a multiset of numbers:

```
fmod NAT-MSET-MIN is
  sorts Nat NatMSet .
  subsort Nat < NatMSet .
  op 0 : -> Nat [ctor] .
  op s : Nat -> Nat [ctor] .
  op _ _ : NatMSet NatMSet -> NatMSet [assoc comm ctor] .
  op _<_ : Nat Nat -> Bool .
  op min : NatMSet -> Nat .
  vars N M : Nat .
  var  S : NatMSet .
  eq 0 < s(N) = true .
  eq s(N) < 0 = false .
  eq s(N) < s(M) = N < M .
  eq min(N N S) = min(N S) .
  ceq min(N M S) = min(N S) if N < M .
  ceq min(N M) = N if N < M .
  eq min(N) = N .
endfm
```

   This module has *two* functions that are not completely defined, that is, such that when invoked on some ground terms do not reduce to constructors. (**Note**: for the sort `Bool`, the constructors are of course `true` and `false`). Do the following:

   - identify the two functions that fail to be fully defined by evaluating them on suitable ground term arguments.
   - correct the specification by adding some extra equations (without modifying the present ones) so that all the functions in the module become completely defined.

   Include both a screenshot of your evaluation of problematic expressions in the original module as well as the correct specification in your answer. Also, send a file with the correct module to `meseguer@cs.uiuc.edu`. You can retrieve the module itself from the course web page.

2. Define a module of lists of quoted identifiers where the list concatenation operation is a constructor modulo associativity and identity and with functions:

```
fmod QIDLIST is protecting QID .
  sort List .
  subsort Qid < List .
  op nil : -> List [ctor] .
  op _ _ : List List -> List [assoc id: nil ctor] .
  op rev : List -> List .
```

```
  op _in_ : Qid List -> Bool .
  op pal : List -> Bool .

  *** add your equations here

endfm
```

You are asked to add ground confluent and terminating equations (modulo the axioms) to the above module defining the non-constructor functions `rev`, `in`, and `pal`, where:

- `rev` is the function that reverses a list, for example, $rev('a\ 'b\ 'c) = 'c\ 'b\ 'a$
- `in` is a predicate that is true of a quoted identifier i and a list l if i appears somewhere in the list l, and is false otherwise
- `pal` is a predicate that is true of a list l if it is a *palindrome* (that is, it reads the same forwards and backwards) and is false otherwise. For example, $pal('a\ 'b\ 'a) = true$, and $pal('a\ 'b\ 'c) = false$.

Your aim should be to define these functions *as simply and elegantly as possible*, taking advantage of the associativity and identity axioms. In particular, you should try to define `rev` with only two equations, `in` also with only two equations and `pal` with only four equations (**Hint:** for defining `in` and `pal` more easily, you can use Maude's `owise` feature). You *do not need any other auxiliary functions*, so using them would be a clear sign that you do not understand the problem.

3. Define a module of multisets of quoted identifiers where the multiset union operation is a constructor modulo associativity and identity (with the empty multiset `mt` as the identity element) and with functions:

```
fmod QIDMSET is protecting QID .
  sort MSet .
  subsort Qid < MSet .
  op mt : -> MSet [ctor] .
  op _ _ : MSet MSet -> MSet [assoc comm id: mt ctor] .
  op _-_ : MSet MSet -> MSet .
  op _in_ : Qid MSet -> Bool .
  op set : MSet -> Bool .

  *** add your equations here

endfm
```

You are asked to add ground confluent and terminating equations (modulo the axioms) to the above module defining the non-constructor functions `-`, `in`, and `set`, where:

- `-` is the binary function that substracts a multiset from another, but keeping track of repeated elements, so that elements are substracted one-by-one; for example, $('c\ 'a\ 'a\ 'b\ 'b\ 'b\ 'b) - ('d\ 'a\ 'a\ 'a\ 'b\ 'b) = 'c\ 'b\ 'b$
- `in` is a predicate that is true of a quoted identifier i and a multiset m if i appears as an element of m, and is false otherwise
- `set` is a predicate that is true of a multiset m if it has no repeated elements and is false otherwise. For example, $set('a\ 'b\ 'a) = false$, and $set('a\ 'b\ 'c) = true$.

Your aim should be to define these functions *as simply and elegantly as possible*, taking advantage of the associativity, commutativity and identity axioms. In particular, you should try to define `-` with only two equations, `in` also with only two equations and `set` with only three equations (**Hint:** for defining these functions more easily, you can use Maude's `owise` feature). You *do not need any other auxiliary functions*, so using them would be a clear sign that you do not understand the problem.

4. Solve Exercise 89 (pages 117–118) in Peter Ölveczky's Lecture Notes. Include your solution and evaluation of some test cases in the hard copy, but email also the code and test cases to `meseguer@cs.uiuc.edu`.