CS 476 Homework #9 Due 10:45am on 10/27

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

1. Consider the following two Maude programs, one whose functions induct on the *left* of a string, and a similar module where they induct on the right of the string (both protect the `PEANO+AC` module also included below):

```
set include BOOL off .

fmod PEANO+AC is sort Nat .
    op 0 : -> Nat [ctor metadata "0"] .
    op s : Nat -> Nat [ctor metadata "4"] .
    op _+_ : Nat Nat -> Nat [assoc comm metadata "8"] .
    vars N M : Nat .
    eq N + 0 = N .
    eq N + s(M) = s(N + M) .
endfm

set include BOOL off .

fmod NAT-LIST+AC is protecting PEANO+AC .
    sorts  NeList List .  subsort Nat < NeList < List .
    op nil : -> List [ctor metadata "1"] .
    op _;_ : List List -> List [assoc metadata "5"] .
    op _;_ : NeList NeList -> NeList [ctor assoc metadata "5"] .
    op rev : List -> List [metadata "10"] .   *** list reverse
    op + : List -> Nat [metadata "12"] .      *** adds all numbers in list
    var N : Nat .  vars L : List .
    eq L ; nil = L .                  eq nil ; L = L .
    eq rev(nil) = nil .
    eq rev(N) = N .                   eq rev(N ; L) = rev(L) ; N .
    eq +(nil) = 0 .   eq +(N) = N .  eq +(N ; L) = N + +(L) .
endfm

set include BOOL off .

fmod NAT-LIST+AC-R is protecting PEANO+AC .
    sorts  List NeList .  subsorts Nat < NeList < List .
    op nil : -> List [ctor metadata "1"] .
    op _;_ : List List -> List [assoc metadata "5"] .
    op _;_ : NeList NeList -> NeList [ctor assoc metadata "5"] .
    op rev : List -> List [metadata "10"] .   *** list reverse
    op + : List -> Nat [metadata "12"] .      *** adds all numbers in list
    var N : Nat .  vars L : List .
    eq L ; nil = L .                  eq nil ; L = L .
    eq rev(nil) = nil .
    eq rev(N) = N .                   eq rev(L ; N) = N ; rev(L) .
    eq +(nil) = 0 .   eq +(N) = N .  eq +(L ; N) = +(L) + N .
endfm
```

Do the following:

(a) Check that both `NAT-LIST+AC` and `NAT-LIST+AC-R` are AvAC-RPO terminating, sort-decreasing, locally confluent, and sufficiently complete (and therefore *admissible* as Maude programs) using the MTA, Church-Rosser Checker, and SCC tools. Make sure to send to `nishant2@illinois.edu` *screenshots* of all your tool interactions.

(b) Use the NuITP to prove that `NAT-LIST+AC` and `NAT-LIST+AC-R` are *semantically equivalent* equational programs, i.e., that they are related by the $\_ \equiv_{sem} \_$ program equivalence relation, so that they have the same canonical term algebra and therefore define the *same* mathematical functions on strings. Again, make sure to send to `nishant2@illinois.edu` a *screenshots* of all your NuITP interactions.

Note: In both using the MTA tool and later when using the NuITP, you may get some strange warning of the form:

```
Warning: constructor declarations for operator _;_^AC failed constructor
    consistency check on 16 out of 16 sort tuples. First such tuple is ([
    List], [List]).
```

You should utterly *disregard* such a warning. It is due to a known bug in the aacrpo.maude program used by both MTA and the NuITP. This bug will be fixed in later NuITP alphas.

2. Let $(\Sigma_L, E_L \cup B)$ and $(\Sigma_R, E_R \cup B)$ denote the equational theories of, respectively, `NAT-LIST+AC` and `NAT-LIST+AC-R`. You have just proved in Problem 1-(b) that `fmod`$(\Sigma_L, E_L \cup B)$`endfm` $\equiv_{sem}$ `fmod`$(\Sigma_R, E_R \cup B)$`endfm`. In this second problem you are asked to *prove* that $(\Sigma_L, E_L \cup B)$ and $(\Sigma_R, E_R \cup B)$ are *not* equivalent equational theories, i.e., that $(\Sigma_L, E_L \cup B) \not\equiv (\Sigma_R, E_R \cup B)$.