Homework XIII

1. [20 points]
This problem pertains to a module to specify a highly simplified version of the (positive and negative) integers. It closely resembles the module for natural numbers in our text (p. 449). The positive integers are obtained by repeated application of the 'succ' function to 0, and the negative integers are obtained by applications of the 'pred' function to 0.

```
module Integers
  exports
    sorts Integer
    operations
      0: Integer
      succ(_): Integer → Integer
      pred(_): Integer → Integer
      add: Integer, Integer → Integer
      sub: Integer, Integer → Integer
  end exports
  variables
    m,n: Integer
  equations
    add(m,0) = m
    add(m,succ(n)) = succ(add(m,n))
    add(m,pred(n)) = pred(add(m,n))
    succ(pred(m)) = m
    pred(succ(m)) = m
end Integers
```

The specification provided above is incomplete as it does not prescribe any behavior for the subtraction function 'sub'. Provide suitable equations for the 'sub' function, and justify that your equations permit the usual conclusions about integer arithmetic (e.g., m-m=0).

2. [30 points]
This problem pertains to the module 'Lists' on page 453 of our text. Add the specification of two functions with the signatures

drop: Natural, List → List, and
front: List → List.

The informal description of the 'drop' function is that drop(N,L) consists of the list L with the first N items removed. For instance,
drop(succ(1), cons(0, cons(1, cons(1, mkList(0))))) ≡ cons(1, mkList(0)).

The informal description of the 'front' function is that front(L) consists of the list L with its last item omitted. For instance,
front(cons(0, cons(1, cons(1, mkList(0))))) ≡ cons(0, cons(1, mkList(1))).

Provide equations to specify these two functions, and justify that your specification accomplishes the informally stated behavior, including when the lists are “too short”.

3. [15 points]

An abridged version of the Stack of Nat ADT with sorts Stack and Nat is given as follows:

Signature	Equations (for all x,y∈Nat and z∈Stack)
0:Nat	
s: Nat → Nat	sum(0,y) = y
sum: Nat, Nat → Nat	sum(s(x), y) = s(sum(x,y))
empty: Stack	
push: Stack, Nat → Stack	
pop: Stack → Stack	pop(push(z, x)) = z
top: Stack → Nat	top(push(z, x)) = x

Give a context-free grammar (BNF) whose derivation trees are (when viewed as terms) precisely the set of well-formed ground terms of this ADT, and justify it.