```
qdata Coin      = {Heads | Tails}
qdata NList a  = {Cens (a, NList(a))  | Nel (a)}

toss ::( ; c:Coin) =
{  q = |0>;
   Had q;
   measure q of
     |0> => {c = Heads}
     |1> => {c = Tails}
}

elect::(lis:NList(Int); leader :Int) = {
  vote lis;
  case lis of
    Cens(h,t) => {
       leader = elect (Cens(h,t))
       }
    Nel(hed) => { leader = hed }
}


vote::(inlis:NList(Int); outlis:NList(Int)) = {
   subvote(inlis ; btoss,tltoss, tres);
   discard tltoss;
   case toss() of
     Heads => { discard btoss; outlis = tres}
     Tails => {
       case btoss of
         Tails => { outlis = tres}
         Heads => {
           case tres of
             Nel (hed) => {outlis = Nel(hed)}
             Cens(hd,tail) => { discard hd; outlis = tail}
         }
     }

}

subvote::(inslis:NList(Int); bToss: Coin, prevToss :Coin, outslis :NList(Int)) ={
case inslis of
Nel(hed)  => {
   outslis = Nel(hed);
   case toss() of
     Heads => { bToss = Heads; prevToss = Heads}
     Tails => { bToss = Tails; prevToss = Tails}
}
Cens(h,tl) => {
  subvote(tl ; bToss, prtoss, tailRes);
  checkToss(toss(),prtoss,h,tailRes; outslis, prevToss);
}
}


checkToss::(t1:Coin, t2:Coin, a:Int, lis:NList(Int);
            outlis:NList(Int), ctoss :Coin) = {
case t1 of
  Heads => {
    case t2 of
      Heads => { outlis = Cens (a, lis); ctoss = Heads}
      Tails => {
         case lis of
```

```
           Nel (b) => { discard b; outlis = Nel(a)}
           Cens(b,tl) => { discard b; outlis = Cens (a,tl)};
         ctoss = Heads
      }
  }
  Tails => { outlis = Cens(a,lis); ctoss = Tails; discard t2;}
}

makeNListOfLength::(i:Int | ; mlis : NList(Int)) = {
  if i == 1 => { mlis = Nel(1) }
  else => {
     ident = i;
     tl = makeNListOfLength(i-1|);
     mlis = Cens( ident, tl)
  }
}
main::() =
{  lis = makeNListOfLength(5|);
   ldr = elect(lis)
}
```